Advances in Large Vocabulary Continuous Speech Recognition

GEOFFREY ZWEIG AND
MICHAEL PICHENY

IBM T.J. Watson Research Center
Yorktown Heights, NY 10598
USA
gzweig@us.ibm.com
picheny@us.ibm.com

Abstract

The development of robust, accurate and efficient speech recognition systems is critical to the widespread adoption of a large number of commercial applications. These include automated customer service, broadcast news transcription and indexing, voice-activated automobile accessories, large-vocabulary voice-activated cell-phone dialing, and automated directory assistance. This article provides a review of the current state-of-the-art, and the recent research performed in pursuit of these goals.

ISSN: 0065-2458/DOI 10.1016/S0065-2458(03)60007-0

1. Introduction

Over the course of the past decade, automatic speech recognition technology has advanced to the point where a number of commercial applications are now widely deployed and successful: systems for name-dialing [84,26], travel reservations [11, 72], getting weather-information [97], accessing financial accounts [16], automated directory assistance [41], and dictation [86,9,78] are all in current use. The fact that these systems work for thousands of people on a daily basis is an impressive testimony to technological advance in this area, and it is the aim of this article to describe the technical underpinnings of these systems and the recent advances that have made them possible. It must be noted, however, that even though the technology has matured to the point of commercial usefulness, the problem of large vocabulary continuous speech recognition (LVCSR) is by no means solved: background noise, corruption by cell-phone or other transmission channels, unexpected shifts in topic, foreign accents, and overly casual speech can all cause automated systems to fail. Thus, where appropriate, we will indicate the shortcomings of current technology, and suggest areas of future research. Although this article aims for a fairly comprehensive coverage of today's speech recognition systems, a vast amount of work has been done in this area, and some limitation is necessary. Therefore, this review will focus primarily on techniques that have proven successful to the point where they have been widely adopted in competition-grade systems such as [78,36,37,58,27,93].

The cornerstone of all current state-of-the-art speech recognition systems is the Hidden Markov Model (HMM) [6,43,54,74]. In the context of HMMs, the speech recognition problem is decomposed as follows. Speech is broken into a sequence of acoustic observations or frames, each accounting for around 25 milliseconds of speech; taken together, these frames comprise the acoustics **a** associated with an utterance. The goal of the recognizer is to find the likeliest sequence of words **w** given the acoustics:

$$\arg\max_{\mathbf{w}} P(\mathbf{w}|\mathbf{a}).$$

This can then be rewritten as:

$$\arg\max_{\mathbf{w}} P(\mathbf{w}|\mathbf{a}) = \arg\max_{\mathbf{w}} \frac{P(\mathbf{w})P(\mathbf{a}|\mathbf{w})}{P(\mathbf{a})}.$$

Since the prior on the acoustics is independent of any specific word hypothesis, the denominator can be ignored, leaving the decomposition:

$$\arg\max_{\mathbf{w}} P(\mathbf{w}|\mathbf{a}) = \arg\max_{\mathbf{w}} P(\mathbf{w})P(\mathbf{a}|\mathbf{w}).$$

The first factor, $P(\mathbf{w})$, is given by the language model, and sets the prior on word sequences. The second factor, $P(\mathbf{a}|\mathbf{w})$ is given by the acoustic model, and links word sequences to acoustics, and is described by an HMM.

The breakdown of a system into acoustic and language model components is one of the main characteristics of current LVCSR systems, and the details of these models are discussed in Sections 3 and 4. However, even with well-defined acoustic and language models that allow for the computation of $P(\mathbf{w})$ and $P(\mathbf{a}|\mathbf{w})$ for any given word and acoustic sequences $\mathbf{w}$ and $\mathbf{a}$, the problem of finding the likeliest single sequence of words remains computationally difficult, and is the subject of a number of specialized search algorithms. These are discussed in Section 5. The final component of current LVCSR systems performs the function of speaker adaptation, and adjusts the acoustic models to match the specifics of an individual voice. These techniques include Maximum A-Posteriori (MAP) adaptation [28], methods that work by adjusting the acoustic features to more closely match generic acoustic models [24], and methods that adjust the acoustic models to match the feature vectors [51]. The field of speaker adaptation has evolved quite dramatically over the past decade, and is currently a key research area; Section 6 covers it in detail. The combination of acoustic and language models, search, and adaptation that characterize current systems is illustrated in Fig. 1.

2. Front End Signal Processing

Currently, there are two main ways in which feature vectors are computed, both motivated by information about human perception. The first of these ways produces features known as *Mel Frequency Cepstral Coefficients* (MFCCs) [17], and the second method is known as *Perceptual Linear Prediction* (PLP) [38]. In both cases, the speech signal is broken into a sequence of overlapping frames which serve as the basis of all further processing. A typical frame-rate is 100 per second, with each frame having a duration of 20 to 25 milliseconds.

After extraction, the speech frames are subjected to a sequence of operations resulting in a compact representation of the perceptually important information in the

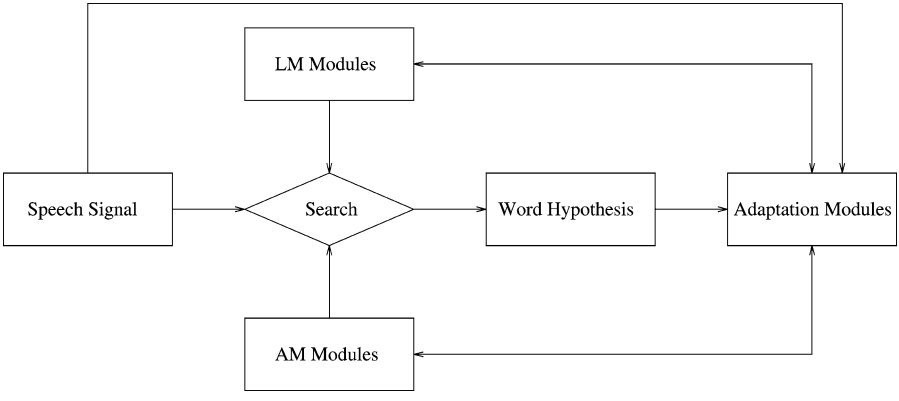

FIG. 1. Sample LVCSR architecture.

speech. Algorithmically, the steps involved in both methods are approximately the same, though the motivations and details are different. In both cases, the algorithmic process is as follows:

(1) compute the power spectrum of the frame,
(2) warp the frequency range of the spectrum so that the high-frequency range is compressed,
(3) compress the amplitude of the spectrum,
(4) decorrelate the elements of the spectral representation by performing an inverse DFT—resulting in a cepstral representation.

Empirical studies have shown that recognition performance can be further enhanced with the inclusion of features computed not just from a single frame, but from several surrounding frames as well. One way of doing this is to augment the feature vectors with the first and second temporal derivatives of the cepstral coefficients [22]. More recently, however, researchers have applied linear discriminant analysis [19] and related transforms to project a concatenated sequence of feature vectors into a low-dimensional space in which phonetic classes are well separated. The following subsections will address MFCCs, PLP features, and discriminant transforms in detail.

2.1 Mel Frequency Cepstral Coefficients

The first step in the MFCC processing of a speech frame is the computation of a short-term power spectrum [17]. In a typical application in which speech is transmitted by phone, it is sampled at 8000 Hz and bandlimited to roughly 3800 Hz. A 25 millisecond frame is typical, resulting in 200 speech samples.This is zero-padded,

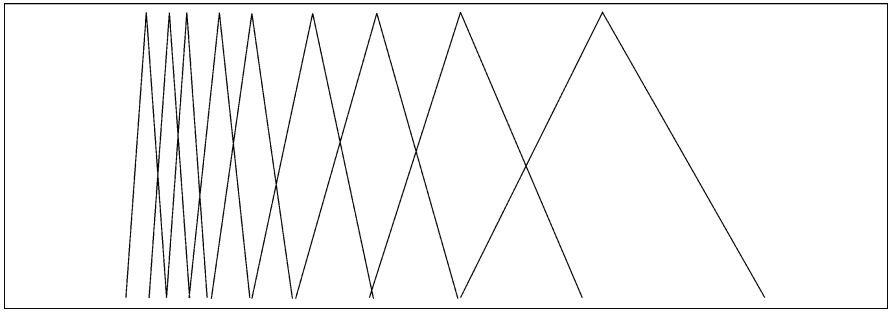

FIG. 2. Mel frequency filters grow exponentially in size.

windowed with the Hamming function

$$W(n) = 0.54 + 0.46\cos\left(\frac{2\pi n}{N-1}\right)$$

and an FFT is used to compute a 128 point power spectrum.

The next step is to compute a warped representation of the power spectrum in which a much coarser representation is used for the high frequencies. This mirrors psychoacoustic observations that human hearing is less precise as frequency increases. To do this, the power spectrum is filtered by a sequence of triangularly shaped filterbanks, whose centers are spaced linearly on the mel scale. The Mel frequency warping [94] is given by

$$f' = 2595\log_{10}\left(1 + \frac{f}{700}\right),$$

so the bandwidth increases exponentially with frequency. Figure 2 illustrates the shape of the mel-frequency filters.[1] Typical applications use 18 to 24 filterbanks spaced between 0 and 4000 Hz [78,46]. This mel frequency warping is similar to the use of critical bands as defined in [100].

After the spectrum is passed through the mel frequency filters, the output of each filter is compressed through the application of a logarithm, and the cepstrum is computed. With F filterbank outputs x_k, the ith MFCC is given by:

$$\text{MFCC}_i = \sum_{k=1}^{F} x_k \cos\left[i\left(k - \frac{1}{2}\right)\frac{\pi}{F}\right], \quad i = 1, 2, \ldots, F.$$

In a typical implementation, the first 13 cepstral coefficients are retained.

MFCCs have the desirable property that linear channel distortions can to some extent be removed through mean subtraction. For example, an overall gain applied to

[1]The original paper [17] used fixed-width filters below 1000 Hz.

the original signal will be removed through mean-subtraction, due to the logarithmic nonlinearity. Therefore, mean-subtraction is standard.

2.2 Perceptual Linear Predictive Coefficients

Perceptual Linear Prediction is similar in implementation to MFCCs, but different in motivation and detail. In practice, these differences have proved to be important, both in lowering the overall error rate, and because PLP-based systems tend to make errors that are somewhat uncorrelated with those in MFCC systems. Therefore, as discussed later in Section 5.4, multiple systems differing in the front-end and other details can be combined through voting to reduce the error rate still further.

The principal differences between MFCC and PLP features are:

- The shape of the filterbanks.
- The use of equal-loudness preemphasis to weight the filterbank outputs.
- The use of cube-root compression rather than logarithmic compression.
- The use of a (parametric) linear-predictive model to determine cepstral coefficients, rather than the use of a (non-parametric) discrete cosine transform.

The first step in PLP analysis is the computation of a short-term spectrum, just as in MFCC analysis. The speech is then passed through a sequence of filters that are spaced at approximately one-Bark intervals, with the Bark frequency Ω being related to un-warped frequency ω (in rad/s) by:

$$\Omega(\omega) = 6\log\left\{\omega/1200\pi + \left[(\omega/1200\pi)^2 + 1\right]^{0.5}\right\}.$$

The shape of the filters is trapezoidal, rather than triangular, motivated by psychophysical experiments [79,101].

Conceptually, after the filterbank outputs are computed, they are subjected to equal-loudness preemphasis. A filterbank centered on (unwarped) frequency ω is modulated by

$$E(\omega) = \left[\left(\omega^2 + 56.8 \times 10^6\right)\omega^4\right] / \left[\left(\omega^2 + 6.3 \times 10^6\right)^2 \times \left(\omega^2 + 0.38 \times 10^9\right)\right].$$

This reflects psycho-physical experiments indicating how much energy must be present in sounds at different frequencies in order for them to be perceived as equally loud. In practice, by appropriately shaping the filters, this step can be done simultaneously with the convolution that produces their output. The weighted outputs are then cube-root compressed, $o' = o^{0.33}$.

In the final PLP step, the warped spectrum is represented with the cepstral coefficients of an all-pole linear predictive model [56]. This is similar to the DCT operation in MFCC computation, but the use of an all-pole model makes the results

more sensitive to spectral peaks, and smooths low-energy regions. In the original implementation of [38], a fifth-order autoregressive model was used; subsequent implementations use a higher order model, e.g., 12 as in [46].

2.3 Discriminative Feature Spaces

As mentioned earlier, it has been found that improved performance can be obtained by augmenting feature vectors with information from surrounding frames [22]. One relatively simple way of doing this is to compute the first and second temporal derivatives of the cepstral coefficients; in practice, this can be done by appending a number of consecutive frames (nine is typical) and multiplying with an appropriate matrix.

More recently [35,88], it has been observed that pattern recognition techniques might be applied to transform the features in a way that is more directly related to reducing the error rate. In particular, after concatenating a sequence of frames, linear discriminant analysis can be applied to find a projection that maximally separates the phonetic classes in the projected space.

Linear discriminant analysis proceeds as follows. We will denote the class associated with example i as $c(i)$. First, the means μ_j and covariances Σ_j of each class are computed, along with the overall mean μ and variance Σ:

$$\mu_j = \frac{1}{N_j} \sum_{i \text{ s.t. } c(i)=j} \mathbf{x}_i, \quad \Sigma_j = \frac{1}{N_j} \sum_{i \text{ s.t. } c(i)=j} (\mathbf{x}_i - \boldsymbol{\mu}_j)(\mathbf{x}_i - \boldsymbol{\mu}_j)^{\mathrm{T}},$$

$$\mu = \frac{1}{N} \sum_i \mathbf{x}_i, \quad \Sigma = \frac{1}{N} \sum_i (\mathbf{x}_i - \boldsymbol{\mu})(\mathbf{x}_i - \boldsymbol{\mu})^{\mathrm{T}}.$$

Next, the total within class variance W is computed

$$W = \frac{1}{N} \sum_j N_j \Sigma_j.$$

Using θ to denote the LDA transformation matrix, the LDA objective function is given by:

$$\hat{\theta} = \arg\max_{\theta} \frac{|\theta^{\mathrm{T}} \Sigma \theta|}{|\theta^{\mathrm{T}} W \theta|},$$

and the optimal transform is given by the top eigenvectors of $W^{-1}\Sigma$.

While LDA finds a projection that tends to maximize relative interclass distances, it makes two questionable assumptions: first, that the classes are modeled by a full covariance Gaussian in the transformed space, and second that the covariances of

all transformed classes are identical. The first assumption is problematic because, as discussed in Section 3.3, full covariance Gaussians are rarely used; but the extent to which the first assumption is violated can be alleviated by applying a subsequent transformation meant to minimize the loss in likelihood between the use of full and diagonal covariance Gaussians [31]. The MLLT transform developed in [31] applies the transform ψ that minimizes

$$\sum_j N_j \left(\log\left|\text{diag}\left(\psi \Sigma_j \psi^{\mathrm{T}}\right)\right| - \log\left|\psi \Sigma_j \psi^{\mathrm{T}}\right|\right)$$

and has been empirically found to be quite effective in conjunction with LDA [77].

To address the assumption of equal covariances, [77] proposes the maximization of

$$\prod_j \left(\frac{|\theta \Sigma \theta^{\mathrm{T}}|}{|\theta \Sigma_j \theta^{\mathrm{T}}|}\right)^{N_j}$$

and presents favorable results when used in combination with MLLT. A closely related technique, HLDA, [50] relates projective discriminant analysis to maximum likelihood training, where the unused dimensions are modeled with a shared covariance. This form of analysis may be used both with and without the constraint that the classes be modeled by a diagonal covariance model in the projected space, and has also been widely adopted. Combined, LDA and MLLT provide on the order of a 10% relative reduction in word-error rate [77] over simple temporal derivatives.

3. The Acoustic Model

3.1 Hidden Markov Model Framework

The job of the acoustic model is to determine word-conditioned acoustic probabilities, $P(\mathbf{a}|\mathbf{w})$. This is done through the use of Hidden Markov Models, which model speech as being produced by a speaker whose vocal tract configuration proceeds through a sequence of states, and produces one or more acoustic vectors in each state. An HMM consists of a set of states $\mathcal{S}$, a set of acoustic observation probabilities, $b_j(o)$, and a set of transition probabilities a_{ij}. The transition and observation probabilities have the following meaning:

(1) $b_j(o)$ is a function that returns the probability of generating the acoustic vector o in state j. $b_j(o_t)$ is the probability of seeing the specific acoustics associated with time t in state j. The observation probabilities are commonly modeled with Gaussian mixtures.
(2) a_{ij} is the time-invariant probability of transitioning from state i to state j.

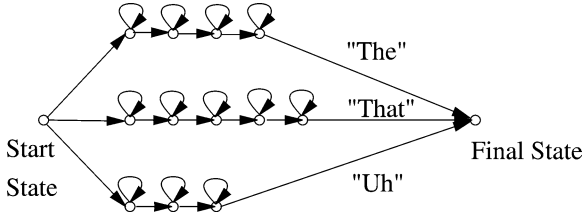

FIG. 3. A simple HMM representing the state sequence of three words. Adding an arc from the final state back to the start state would allow repetition.

Note that in the HMM framework, each acoustic vector is associated with a specific state in the HMM. Thus, a sequence of n acoustic vectors will correspond to a sequence of n consecutive states. We will denote a specific sequence of states $s_1 = a,\ s_2 = b,\ s_3 = c,\ \ldots,\ s_n = k$ by $\mathbf{s}$. In addition to normal emitting states, it is often convenient to use "null" states, which do not emit acoustic observations. In particular, we will assume that the HMM starts at time $t = 0$ in a special null start-state α, and that all paths must end in a special null final-state ω at $t = N + 1$. In general, having a specific word hypothesis $\mathbf{w}$ will be compatible with only some state sequences, $\mathbf{s}$, and not with others. It is necessary, therefore, to constrain sums over state sequences to those sequences that are compatible with a given word sequence; we will not, however, introduce special notation to make this explicit. With this background, the overall probability is factored as follows:

$$P(\mathbf{a}|\mathbf{w}) = \sum_{\mathbf{s}} P(\mathbf{a}|\mathbf{s})P(\mathbf{s}|\mathbf{w}) = \sum_{\mathbf{s}} \prod_{t=1,\ldots,n} b_{s_t}(o_t)a_{s_t s_{t-1}}.$$

Figure 3 illustrates a simple HMM that represents the state sequences of three words.

The following sections describe the components of the HMM acoustic model in more detail. Section 3.2 will focus on the mapping from words to states that is necessary to determine $P(\mathbf{s}|\mathbf{w})$. Section 3.3 discusses the Gaussian mixture models that are typically used to model $b_j(o)$. The transition probabilities can be represented in a simple table, and no further discussion is warranted. The section closes with a description of the training algorithms used for parameter estimation.

3.2 Acoustic Context Models

In its simplest form, the mapping from words to states can be made through the use of a phonetic lexicon that associates one or more sequences of phonemes with each word in the vocabulary. For example,

```
barge    |     B AA R JH
tomato   |     T AH M EY T OW
tomato   |     T AH M AA T OW
```

Typically, a set of 40 phonemes is used, and comprehensive dictionaries are available [14,15].

In practice, coarticulation between phones causes this sort of invariant mapping to perform poorly, and instead some sort of context-dependent mapping from words to acoustic units is used [95,5]. This mapping takes each phoneme and the phonemes that surround it, and maps it into an acoustic unit. Thus, the "AA" in "B AA R JH" may have a different acoustic model than the "AA" in "T AH M AA T OW." Similarly, the "h" in "hammer" may be modeled with a different acoustic unit depending on whether it is seen in the context of "the hammer" or "a hammer." The exact amount of context that is used can vary, the following being frequently used:

(1) Word-internal triphones. A phone and its immediate neighbors to the left and right. However, special units are used at the beginnings and endings of words so that context does not persist across word boundaries.
(2) Cross-word triphones. The same as above, except that context persists across word boundaries, resulting in better coarticulation modeling.
(3) Cross-word quinphones. A phone and its two neighbors to the left and right.
(4) A phone, and all the other phones in the same word.
(5) A phone, all the other phones in the same word, and all phones in the preceding word.

When a significant amount of context is used, the number of potential acoustic states becomes quite large. For example, with triphones the total number of possible acoustic models becomes approximately $40^3 = 64{,}000$. In order to reduce this number, decision-tree clustering is used to determine equivalence classes of phonetic contexts [5,95]. A sample tree is shown in Fig. 4. The tree is grown in a top-down fashion using an algorithm similar to that of Fig. 5. Thresholds on likelihood gain, frame-counts, or the Bayesian information criterion [10] can be used to determine an appropriate tree depth.

In a typical large vocabulary recognition system [78], it is customary to have a vocabulary size between 30 and 60 thousand words and two or three hundred hours of training data from hundreds of speakers. The resulting decision trees typically have between 4000 and 12,000 acoustic units [78,46].

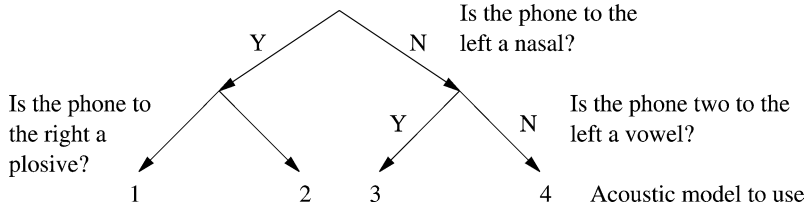

FIG. 4. Decision tree for clustering phonetic contexts.

1. Create a record for each frame that includes the frame and the phonetic context associated with it.
2. Model the frames associated with a node with a single diagonal-covariance Gaussian. The frames associated with a node will have a likelihood according to this model.
3. For each yes/no question based on the context window, compute the likelihood that would result from partitioning the examples according to the induced split.
4. Split the frames in the node using the question that results in the greatest likelihood gain, and recursively process the resulting two nodes.

FIG. 5. Decision tree building.

3.3 Gaussian Mixture State Models

The observation probabilities $b_j(o)$ are most often modeled with mixtures of Gaussians. The likelihood of the d-dimensional feature vector $\mathbf{x}$ being emitted by state j is given by:

$$b_j(\mathbf{x}) = \sum_k m_{jk} \left((2\pi)^d |\Sigma_{jk}| \right)^{-1/2} \exp\left(-\frac{1}{2} (\mathbf{x} - \boldsymbol{\mu}_{jk})^{\mathrm{T}} \Sigma_{jk}^{-1} (\mathbf{x} - \boldsymbol{\mu}_{jk}) \right)$$

where the coefficients m_{jk} are mixture weights, $\sum_k m_{jk} = 1$. This can be expressed more compactly as

$$b_j(\mathbf{x}) = \sum_k m_{jk} \mathcal{N}(\mathbf{x}; \boldsymbol{\mu}_{jk}, \Sigma_{jk}).$$

In order to minimize the amount of computation required to compute observation probabilities, it is common practice to use diagonal covariance matrices. Between 150,000 and 300,000 Gaussians are typical in current LVCSR systems.

The use of diagonal covariance matrices has proved adequate, but requires that the dimensions of the feature vectors be relatively uncorrelated. While the linear transforms described in Section 2 can be used to do this, recently there has been a significant amount of work focused on more efficient covariance representations.

One example of this is EMLLT [70], in which the inverse covariance matrix of each Gaussian j is modeled as the sum of basis matrices. First, a set of d dimensional basis vectors a_l is defined. Then inverse covariances are modeled as:

$$\Sigma_j^{-1} = \sum_{l=1}^{D} \lambda_l^j \mathbf{a}_l \mathbf{a}_l^{\mathrm{T}}.$$

One of the main contributions of [70] is to describe a maximum-likelihood training procedure for adjusting the basis vectors. Experimental results are presented that show improved performance over both diagonal and full-covariance modeling in a recognition system for in-car commands. In further work [3], this model has been generalized to model both means and inverse-covariance matrices in terms of basis expansions (SPAM).

3.4 Maximum Likelihood Training

A principal advantage of HMM-based systems is that it is quite straightforward to perform maximum likelihood parameter estimation. The main step is to compute posterior state-occupancy probabilities for the HMM states. To do this, the following quantities are defined:

- $\alpha_j(t)$: the probability of the observation sequence up to time t, and accounting for o_t in state j.
- $\beta_j(t)$: the probability of the observation sequence $o_{t+1} \ldots o_N$ given that the state at time t is j.
- $P = \sum_k \alpha_k(t)\beta_k(t)$: the total data likelihood, constant over t.
- $\gamma_j(t) = \frac{\alpha_j(t)\beta_j(t)}{\sum_k \alpha_k(t)\beta_k(t)}$: the posterior probability of being in state j at time t.
- $\mathrm{mix}_{jk}(t) = \frac{m_{jk}\mathcal{N}(\mathbf{o}_t; \boldsymbol{\mu}_{jk}, \boldsymbol{\Sigma}_{jk})}{\sum_f m_{jf}\mathcal{N}(\mathbf{o}_t; \boldsymbol{\mu}_{jf}, \boldsymbol{\Sigma}_{jf})}$: the probability of mixture component k given state j at time t.

The α and β quantities can be computed with a simple recursion:

- $\alpha_j(t) = \sum_i \alpha_i(t-1)a_{ij}b_j(o_t)$.
- $\beta_j(t) = \sum_k a_{jk}b_k(o_{t+1})\beta_k(t+1)$.

The recursions are initialized by setting all αs and βs to 0 except:

- $\alpha_\alpha(0) = 1$.
- $\beta_\omega(N+1) = 1$.

Once the posterior state occupancy probabilities are computed, it is straightforward to update the model parameters for a diagonal-covariance system [54,94,73].

- $\hat{a}_{ij} = \frac{\sum_t \alpha_i(t) a_{ij} b_j(o_{t+1}) \beta_j(t+1)}{\sum_t \alpha_i(t)\beta_i(t)}$.
- $\hat{\mu}_{jk} = \frac{\sum_t \gamma_j(t) \text{mix}_{jk}(t) \mathbf{o}_t}{\sum_t \gamma_j(t) \text{mix}_{jk}(t)}$.
- $\hat{\Sigma}_{jk} = \frac{\sum_t \gamma_j(t) \text{mix}_{jk}(t) (\mathbf{o}_t - \hat{\mu}_{jk})(\mathbf{o}_t - \hat{\mu}_{jk})^{\mathrm{T}}}{\sum_t \gamma_j(t) \text{mix}_{jk}(t)}$.

This discussion has avoided a number of subtleties that arise in practice, but are not central to the ideas. Specifically, when multiple observation streams are available, an extra summation must be added outside all others in the reestimation formulae. Also, observation probabilities are tied across multiple states—the same "ae" acoustic model may be used in multiple HMM states. This entails creating summary statistics for each acoustic model by summing the statistics of all the states that use it. Finally, in HMMs with extensive null states, the recursions and reestimation formulae must be modified to reflect the spontaneous propagation of probabilities through chains of null states.

3.4.1 Maximum Mutual Information Training

In standard maximum likelihood training, the model parameters for each class are adjusted in isolation, so as to maximize the likelihood of the examples of that particular class. While this approach is optimal in the limit of infinite training data [62], it has been suggested [63,4] that under more realistic conditions, a better training objective might be to maximize the amount of mutual information between the acoustic vectors and the word labels. That is, rather than training so as to maximize

$$P_\theta(\mathbf{w}, \mathbf{a}) = P_\theta(\mathbf{w}) P_\theta(\mathbf{a}|\mathbf{w})$$

with respect to θ, to train so as to maximize

$$\sum_{\mathbf{w},\mathbf{a}} P_\theta(\mathbf{w}, \mathbf{a}) \log \frac{P_\theta(\mathbf{w}, \mathbf{a})}{P_\theta(\mathbf{w}) P_\theta(\mathbf{a})}.$$

Using the training data D to approximate the sum over all words and acoustics, we can represent the mutual information as

$$\sum_D \log \frac{P_\theta(\mathbf{a}, \mathbf{w})}{P_\theta(\mathbf{w}) P_\theta(\mathbf{a})} = \sum_D \log \frac{P_\theta(\mathbf{w}) P_\theta(\mathbf{a}|\mathbf{w})}{P_\theta(\mathbf{w}) P_\theta(\mathbf{a})} = \sum_D \log \frac{P_\theta(\mathbf{a}|\mathbf{w})}{P_\theta(\mathbf{a})}$$

$$= \sum_D \log \frac{P_\theta(\mathbf{a}|\mathbf{w})}{\sum_{\mathbf{w}'} P_\theta(\mathbf{w}') P_\theta(\mathbf{a}|\mathbf{w}')}.$$

If we assume that the language model determining $P_\theta(\mathbf{w})$ is constant (as is the case in acoustic model training) then this is identical to optimizing the posterior word probability:

$$\sum_D \log P_\theta(\mathbf{w}|\mathbf{a}) = \sum_D \log \frac{P_\theta(\mathbf{a}|\mathbf{w})P_\theta(\mathbf{w})}{\sum_{\mathbf{w}'} P_\theta(\mathbf{w}')P_\theta(\mathbf{a}|\mathbf{w}')}.$$

Before describing MMI training in detail, we note that the procedure that will emerge is not much different from training an ML system. Procedurally, one first computes the state-occupancy probabilities and first and second order statistics exactly as for a ML system. This involves summing path posteriors over all HMM paths that are consistent with the known word hypotheses. One then repeats exactly the same process, but sums over all HMM paths without regard to the transcripts. The two sets of statistics are then combined in a simple update procedure. For historical reasons, the first set of statistics is referred to as "numerator" statistics and the second (unconstrained) set as "denominator" statistics.

An effective method for performing MMI optimization was first developed in [30] for the case of discrete hidden Markov models. The procedure of [30] works in general to improve objective functions $R(\theta)$ that are expressible as

$$R(\theta) = \frac{s_1(\theta)}{s_2(\theta)}$$

with s_1 and s_2 being polynomials with $s_2 > 0$. Further, for each individual probability distribution λ under adjustment, it must be the case that $\lambda_i \geqslant 0$ and $\sum_i \lambda_i = 1$. In this case, it is proved that the parameter update

$$\hat{\lambda}_i = \frac{\lambda_i\left(\frac{\partial \log R(\lambda)}{\partial \lambda_i} + D\right)}{\sum_k \lambda_k\left(\frac{\partial \log R(\lambda)}{\partial \lambda_k} + D\right)}$$

is guaranteed to increase the objective function, with a large enough value of the constant D. In the case of discrete variables, it is shown that

$$\frac{\partial \log R(\lambda)}{\partial \lambda_i} = \frac{1}{\lambda_i}\left(C_{\lambda_i}^{\text{num}} - C_{\lambda_i}^{\text{den}}\right)$$

where λ_i is probability of event associated with λ_i being true, and C_{λ_i} is count of times this event occurred, as computed from the α–β recursions of the previous section.

Later work [68,92], extended these updates to Gaussian means and variances, and [92] did extensive work to determine appropriate values of D for large vocabulary speech recognition. For state j, mixture component m, let $S(x)$ denote the first order statistics, $S(x^2)$ denote the second order statistics, and C denote the count of the

number of times a mixture component is used. The update derived is

$$\hat{\mu}_{jm} = \frac{S_{jm}^{\text{num}}(x) - S_{jm}^{\text{den}}(x) + D\mu_{jm}}{C_{jm}^{\text{num}} - C_{jm}^{\text{den}} + D},$$

$$\hat{\sigma}_{jm}^2 = \frac{S_{jm}^{\text{num}}(x^2) - S_{jm}^{\text{den}}(x^2) + D(\sigma_{jm}^2 + \mu_{jm}^2)}{C_{jm}^{\text{num}} - C_{jm}^{\text{den}} + D} - \hat{\mu}_{jm}^2.$$

For the mixture weights, let f_{jm} be the mixture coefficient associated with mixture component m of state j. Then

$$\hat{f}_{jm} = \frac{f_{jm}\left(\frac{\partial \log R(\lambda)}{\partial f_{jm}} + D\right)}{\sum_k f_{jk}\left(\frac{\partial \log R(\lambda)}{\partial f_{jk}} + D\right)}$$

with

$$\frac{\partial \log R(\lambda)}{\partial f_{jk}} = \frac{1}{f_{jk}}\left(C_{jk}^{\text{num}} - C_{jk}^{\text{den}}\right).$$

Several alternative ways for reestimating the mixture weights are given in [92].

MMI has been found to give a 5–10% relative improvement in large vocabulary tasks [92], though the advantage diminishes as systems with larger numbers of Gaussians are used [58]. The main disadvantage of MMI training is that the denominator statistics must be computed over all possible paths. This requires either doing a full decoding of the training data at each iteration, or the computation of lattices (see Section 5.2). Both options are computationally expensive unless an efficiently written decoder is available.

4. Language Model

4.1 Finite State Grammars

Finite state grammars [1,39] are the simplest and in many ways the most convenient way of expressing a language model for speech recognition. The most basic way of expressing one of these grammars is as an unweighted regular expression that represents a finite set of recognizable statements. For example, introductions to phone calls in a three-person company might be represented with the expression

```
(Hello | Hi) (John | Sally | Sam)? it's
(John | Sally | Sam)
```

At a slightly higher level, Backus Naur Form [64] is often used for more elaborate grammars with replacement patterns. For example,

```
<SENTENCE> ::= Greeting.
Greeting ::= Intro Name? it's Name.
Intro ::= Hello | Hi.
Name ::= John | Sally | Sam.
```

In fact, BNF is able to represent context free grammars [13]—a broad class of grammars in which recursive rule definitions allow the recognition of some strings that cannot be represented with regular expressions. However, in comparison with regular expressions, context-free grammars have had relatively little affect on ASR, and will not be discussed further.

Many of the tools and conventions associated with regular expressions were developed in the context of computer language compilers, in which texts (programs) were either syntactically correct or not. In this context, there is no need for a notion of how correct a string is, or alternatively what the probability of it being generated by a speaker of the language is. Recall, however, that in the context of ASR, we are interested in $P(\mathbf{w})$, the probability of a word sequence. This can easily be incorporated in to the regular expression framework, simply by assigning costs or probabilities to the rules in the grammar.

Grammars are frequently used in practical dialog applications, where developers have the freedom to design system prompts and then specify a grammar that is expected to handle all reasonable replies. For example, in an airline-reservation application the system might ask "Where do you want to fly to?" and then activate a grammar designed to recognize city names. Due to their simplicity and intuitive nature, these sorts of grammars are the first choice wherever possible.

4.2 N-gram Models

N-gram language models are currently the most widely used LMs in large vocabulary speech recognition. In an N-gram language model, the probability of each word is conditioned on the $n-1$ preceding words:

$$P(\mathbf{w}) = P(w_1)P(w_2|w_1)\cdots P(w_{n-1}|w_1\ldots w_{n-2}) \times \prod_{i=n}^{i=N} P(w_i|w_{i-1}, w_{i-2}, \ldots, w_{i-n+1}).$$

While in principle this model ignores a vast amount of prior knowledge concerning linguistic structure—part-of-speech classes, syntactic constraints, semantic coher-

ence, and pragmatic relevance—in practice, researchers have been unable to significantly improve on it.

A typical large vocabulary system will recognize between 30 and 60 thousand words, and use a 3 or 4-gram language model trained on around 200 million words [78]. While 200 million words seems at first to be quite large, in fact for a 3-gram LM with a 30,000 word vocabulary, it is actually quite small compared to the 27×10^{12} distinct trigrams that need to be represented. In order to deal with this problem of data sparsity, a great deal of effort has been spent of developing techniques for reliably estimating the probabilities of rare events.

4.2.1 Smoothing

Smoothing is perhaps the most important practical detail in building N-gram language models, and these techniques fall broadly into three categories: additive smoothing, backoff models, and interpolated models. The following sections touch briefly on each, giving a full description for only interpolated LMs, which have been empirically found to give good performance on a variety of tasks. The interested reader can find a full review of all these methods in [12].

4.2.1.1 Additive Smoothing. In the following, we will use the compact notation w_x^y to refer to the sequence of words $w_x, w_{x+1} \ldots w_y$, and $c(w_x^y)$ to the number of times (count) that this sequence has been seen in the training data. The maximum-likelihood estimate of $P(w_i|w_{i-n+1}^{i-1})$ is thus given as:

$$P\left(w_i|w_{i-n+1}^{i-1}\right) = \frac{c(w_{i-n+1}^{i})}{c(w_{i-n+1}^{i-1})}.$$

The problem, of course, is that for high-order N-gram models, many of the possible (and perfectly normal) word sequences in a language will not be seen, and thus assigned zero-probability. This is extraordinarily harmful to a speech recognition system, as one that uses such a model will never be able to decode these novel word sequences. One of the simplest ways of dealing with such a problem is to use a set of fictitious or imaginary counts to encode our prior knowledge that all word sequences have some likelihood. In the most basic implementation [42], one simply adds a constant amount δ to each possible event. For a vocabulary of size $|V|$, one then has:

$$P\left(w_i|w_{i-n+1}^{i-1}\right) = \frac{\delta + c(w_{i-n+1}^{i})}{\delta|V| + c(w_{i-n+1}^{i-1})}.$$

The optimal value of δ can be found simply by performing a search so as to maximize the implied likelihood on a set of held-out data. This scheme, while having the virtue of simplicity, tends to perform badly in practice [12].

4.2.1.2 Low-Order Backoff. One of the problems of additive smoothing is that it will assign the same probability to all unseen words that follow a particular history. Thus for example, it will assign the same probability to the sequence "spaghetti western" as to "spaghetti hypanthium," assuming that neither has been seen in the training data. This violates our prior knowledge that more frequently occurring words are more likely to occur, even in previously unseen contexts.

One way of dealing with this problem is to use a backoff model in which one "backs off" to a low order language model estimate to model unseen events. These models are of the form:

$$P\left(w_i|w_{i-n+1}^{i-1}\right) = \begin{cases} \alpha(w_i|w_{i-n+1}^{i-1}) & \text{if } c(w_{i-n+1}^{i}) > 0, \\ \gamma(w_{i-n+1}^{i-1})P(w_i|w_{i-n+2}^{i-1}) & \text{if } c(w_{i-n+1}^{i}) = 0. \end{cases}$$

One example of this is Katz smoothing [45], which is used, e.g., in the SRI language-modeling toolkit [83]. However, empirical studies have shown that better smoothing techniques exist, so we will not present it in detail.

4.2.1.3 Low-Order Interpolation. The weakness of a backoff language model is that it ignores the low-order language model estimate whenever a high-order N-gram has been seen. This can lead to anomalies when some high-order N-grams are seen, and others with equal (true) probability are not. The most effective type of N-gram model uses an interpolation between high and low-order estimates under all conditions. Empirically, the most effective of these is the modified Kneser–Ney language model [12], which is based on [47].

This model makes use of the concept of the number of unique words that have been observed to follow a given language model history at least k times. Define

$$N_k\left(w_{i-n+1}^{i-1}\bullet\right) = \left|\left\{w_i\colon c\left(w_{i-n+1}^{i-1}w_i\right) = k\right\}\right|$$

and

$$N_{k+}\left(w_{i-n+1}^{i-1}\bullet\right) = \left|\left\{w_i\colon c\left(w_{i-n+1}^{i-1}w_i\right) \geqslant k\right\}\right|.$$

The modified Kneser Ney estimate is then given as

$$P\left(w_i|w_{i-n+1}^{i-1}\right) = \frac{c(w_{i-n+1}^{i}) - D(c(w_{i-n+1}^{i}))}{c(w_{i-n+1}^{i-1})} + \gamma\left(w_{i-n+1}^{i-1}\right)P\left(w_i|w_{i-n+2}^{i-1}\right).$$

Defining

$$Y = \frac{n_1}{n_1 + 2n_2}$$

where n_r is the number of n-grams that occur exactly r times, the discounting factors are given by

$$D(c) = \begin{cases} 0 & \text{if } c = 0, \\ 1 - 2Y\frac{n_2}{n_1} & \text{if } c = 1, \\ 2 - 3Y\frac{n_3}{n_2} & \text{if } c = 2, \\ 3 - 4Y\frac{n_4}{n_3} & \text{if } c \geqslant 3. \end{cases}$$

The backoff weights are determined by

$$\gamma\left(w_{i-n+1}^{i-1}\right) = \frac{D_1 N_1(w_{i-n+1}^{i-1}\bullet) + D_2 N_2(w_{i-n+1}^{i-1}\bullet) + D_{3+} N_{3+}(w_{i-n+1}^{i-1}\bullet)}{c(w_{i-n+1}^{i-1})}.$$

This model has been found to slightly outperform most other models and is in use in state-of-the-art systems [78]. Because $D(0) = 0$, this can also be expressed in a backoff form.

4.2.2 Cross-LM Interpolation

In many cases, several disparate sources of language model training data are available, and the question arises: what is the best method of combining these sources? The obvious answer is simply to concatenate all the sources of training data together, and to build a model. This, however, has some serious drawbacks when the sources are quite different in size. For example, in many systems used to transcribe telephone conversations [78,76,93,27], data from television broadcasts is combined with a set of transcribed phone conversations. However, due to its easy availability, there is much more broadcast data than conversational data: about 150 million words compared to 3 million. This can have quite negative effects. For example, in the news broadcast data, the number of times "news" follows the bigram "in the" may be quite high, whereas in conversations, trigrams like "in the car" or "in the office" are much likelier. Because of the smaller amount of data, though, these counts will be completely dwarfed by the broadcast news counts, with the result that the final language model will be essentially identical to the broadcast news model. Put another way, it is often the case that training data for several *styles of speaking* is available, and that the relative amounts of data in each category bears no relationship to how frequently the different styles are expected to be used in real life.

In order to deal with this, it is common to interpolate multiple distinct language models. For each data source k, a separate language model is built that predicts word

probabilities: $P_k(w_i|w_{i-n+1}^{i-1})$. These models are then combined with weighting factors λ_k:

$$P\left(w_i|w_{i-n+1}^{i-1}\right) = \sum_k P_k\left(w_i|w_{i-n+1}^{i-1}\right), \quad \sum_k \lambda_k = 1.$$

For example, in a recent conversational telephony system [78] an interpolation of data gathered from the web, broadcast news data, and two sources of conversational data (with weighting factors 0.4, 0.2, 0.2, and 0.2 respectively) resulted in about a 10% relative improvement over using the largest single source of conversational training data.

4.2.3 *N*-gram Models as Finite State Graphs

While N-gram models have traditionally been treated as distinct from recognition grammars, in fact they are identical, and this fact has been increasingly exploited. One simple way of seeing this is to consider a concrete algorithm for constructing a finite state graph at the HMM state level from an N-gram language model expressed as a backoff language model. This will make use of two functions that act on a word sequence w_j^k:

(1) head(w_j^k) returns the suffix w_{j+1}^k.
(2) tail(w_j^k) returns the prefix w_j^{k-1}.

For a state-of-the-art backoff model, one proceeds as follows:

(1) for each N-gram with history $\mathbf{q}$ and successor word r make a unique state for $\mathbf{q}$, head($\mathbf{q}r$), and tail($\mathbf{q}$),
(2) for each N-gram add an arc from $\mathbf{q}$ to head($\mathbf{q}r$) labeled with r and weighted by the α probability of the backoff model,
(3) for each unique N-gram history $\mathbf{q}$ add an arc from $\mathbf{q}$ to tail($\mathbf{q}$) with the backoff γ associated with $\mathbf{q}$.

To accommodate multiple pronunciations of a given word, one then replaces each word arc with a set of arcs, one labeled with each distinct pronunciation, and multiplies the associated probability with the probability of that pronunciation. For acoustic models in which there is no cross word context, each pronunciation can then be replaced with the actual sequence of HMM states associated with the word; accommodating cross word context is more complex, but see, e.g., [99]. Figure 6 illustrates a portion of an HMM n-gram graph.

We have described the process of expanding a language model into a finite-state graph as a sequence of "search and replace" operations acting on a basic representation at the word level. However, [59,60] have recently argued that the process is

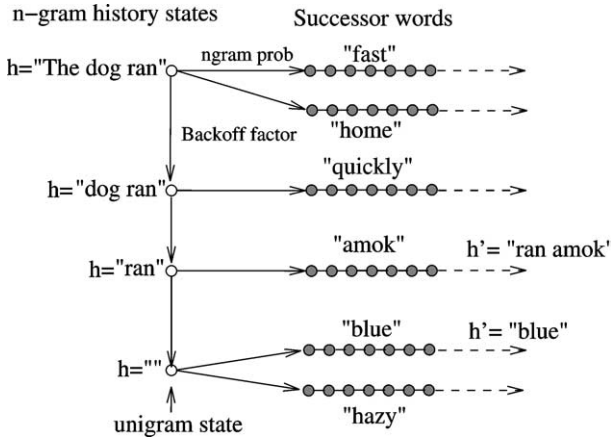

FIG. 6. HMM state graph illustrating the structure of a backoff language model.

best viewed in terms of a sequence of finite state transductions. In this model, one begins with a finite state encoding of the language model, but represents the expansion at each level—from word to pronunciation, pronunciation to phone, and phone to state—as the composition of the previous representation with a finite state transducer. The potential advantage of this approach is a consistent representation of each form of expansion, with the actual operations being performed by a single composition function. In practice, care must be taken to ensure that the composition operations do not use large amounts of memory, and in some cases, it is inconvenient to express the acoustic context model in the form of a transducer (e.g., when long span context models are used).

In some ways, the most important advantage of finite-state representations is that operations of determinization and minimization were recently developed by [59,60]. Classical algorithms were developed in the 1970s [1] for unweighted graphs as found in compilers, but the extension to weighted graphs (the weights being the language model and transition probabilities) has made these techniques relevant to speech recognition. While it is beyond the scope of this paper to present the algorithms for determinization and minimization, we briefly describe the properties.

A graph is said to be deterministic if each outgoing arc from a given state has a unique label. In the context of speech recognition graphs, the arcs are labeled with either HMM states, or word, pronunciation, or phone labels. While the process of taking a graph and finding an equivalent deterministic one is well defined, the deterministic representation can in pathological cases grow exponentially in the number of states of the input graph. In practice, this rarely happens, but the graph does

grow. The benefit actually derives from the specific procedures used to implement the Viterbi search described in Section 5.1. Suppose one has identified a fixed number w of states that are reasonably likely at a given time t. Only a small number k of HMM states are likely to have good acoustic matches, and thus to lead to likely states at time $t+1$. Thus, if on average z outgoing arcs per state are labeled with a given HMM state, the number of likely states at $t+1$ will be on the order of zkw. By using a deterministic graph, z is limited to 1, and thus tends to decrease the number of states that will ever be deemed likely. In practice, this property can lead to an order-of-magnitude speedup in search time, and makes determinization critical.

One can also ask, given a deterministic graph, what is the smallest equivalent deterministic graph. The process of minimization [59] produces such a graph, and in practice often reduces graph sizes by a factor of two or three.

4.2.4 Pruning

Modern corpus collections [33] often contain an extremely large amount of data—between 100 million and a billion words. Given that N-gram language models can backoff to lower-order statistics when high-order statistics are unavailable, and that representing extremely large language models can be disadvantageous from the point-of-view of speed and efficiency, it is natural to ask how one can trade off language model size and fidelity. Probably the simplest way of doing this is to impose a count threshold, and then to use a lower-order backoff estimate for the probability of the nth word in such N-grams.

A somewhat more sophisticated approach [80] looks at the loss in likelihood caused by using the backoff estimate to select N-grams to prune. Using P and P' to denote the original and backed-off estimates, and $N(\cdot)$ to represent the (possibly discounted) number of times an N-gram occurs, the loss in log likelihood caused by the omission of an N-gram w_{i-n+1}^{i} is given by:

$$N\left(w_{i-n+1}^{i}\right)\left(\log P\left(w_i|w_{i-n+1}^{i-1}\right) - \log P\left(w_i|w_{i-n+2}^{i-1}\right)\right).$$

In the "Weighted Difference Method" [80], one computes all these differences, and removes the N-grams whose difference falls below a threshold. A related approach [82] uses the Kullback–Leibler distance between the original and pruned language models to decide which N-grams to prune. The contribution of an N-gram in the original model to this KL distance is given by:

$$P\left(w_{i-n+1}^{i}\right)\left(\log P\left(w_i|w_{i-n+1}^{i-1}\right) - \log P\left(w_i|w_{i-n+2}^{i-1}\right)\right)$$

and the total KL distance is found by summing over all N-grams in the original model. The algorithm of [82] works in batch mode, first computing the change in relative entropy that would result from removing each N-gram, and then removing

all those below a threshold, and recomputing backoff weights. A comparison of the weighted-difference and relative-entropy approaches shows that the two criteria are the same in form, and the difference between the two approaches is primarily in the recomputation of backoff weights that is done in [82]. In practice, LM pruning can be extremely useful in limiting the size of a language model in compute-intensive tasks.

4.2.5 Class Language Models

While n-gram language models often work well, they have some obvious drawbacks, specifically their inability to capture linguistic generalizations. For example, if one knows that the sentence "I went home to feed my dog" has a certain probability, then one might also surmise that the sentence "I went home to feed my cat" is also well-formed, and should have roughly the same probability. There are at least two forms of knowledge that are brought to bear to make this sort of generalization: syntactic and semantic. Syntactically, both "dog" and "cat" are nouns, and can therefore be expected to be used in the same ways in the same sentence patterns. Further, we have the semantic information that both are pets, and this further strengthens their similarity. The importance of the semantic component can be further highlighted by considering the two sentences, "I went home to walk my dog," and "I went home to walk my cat." Here, although the syntactic structure is the same, the second sentence seems odd because cats are not walked.

Class-based language models are an attempt to capture the syntactic generalizations that are inherent in language. The basic idea is to first express a probability distribution over parts-of-speech (nouns, verbs, pronouns, etc.), and then to specify the probabilities of specific instances of the parts of speech. In its simplest form [8] a class based language model postulates that each word maps to a single class, so that the word stream w_i^k induces a sequence of class labels c_i^k. The n-gram word probability is then given by:

$$P\left(w_i|w_{i-n+1}^{i-1}\right) = P(w_i|c_i)P\left(c_i|c_{i-n+1}^{i-1}\right).$$

Operationally, one builds an n-gram model on word classes, and then combines this with a unigram model that specifies the probability of a specific word given a class. This form of model makes the critical assumption that each word maps into a unique class, which of course is not true for standard parts of speech. (For example, "fly" has a meaning both in the verb sense of what a bird does, and in the noun sense of an insect.) However, [8] present an automatic procedure for learning word-classes of this form. This method greedily assigns words to classes so as to minimize the perplexity of induced N-gram model over class sequences. This has the advantage both of relieving the user from specifying grammatical relationships, and of being

able to combine syntactic and semantic information. For example, [8] presents a class composed of *feet miles pounds degrees inches barrels tons acres meters bytes* and many similar classes whose members are similar both syntactically and semantically.

Later work [66] extends the class-based model to the case where a word may map into multiple classes, and a general mapping function $S(\cdot)$ is used to map a word history w_{i-n+1}^{i-1} into a specific equivalence class s. Under these more general assumptions, we have

$$P\left(w_i|w_{i-n+1}^{i-1}\right) = \sum_{c_i} P(w_i|c_i)\left[\sum_s P(c_i|s)P\left(s|w_{i-n+1}^{i-1}\right)\right].$$

Due to the complexity of identifying reasonable word-to-class mappings, however, the class induction procedure presented assumes an unambiguous mapping for each word.

This general approach has been further studied in [67], and experimental results are presented suggesting that automatically derived class labels are superior to the use of linguistic part-of-speech labels. The process can also be simplified [91] to using

$$P\left(w_i|c\left(w_{i-n+1}^{i-1}\right)\right).$$

Class language models are now commonly used in state-of-the-art systems, where their probabilities are interpolated with word-based N-gram probabilities, e.g., [93].

5. Search

Recall that the objective of a decoder is to find the best word sequence $\mathbf{w}^*$ given the acoustics:

$$\mathbf{w}^* = \arg\max_{\mathbf{w}} P(\mathbf{w}|\mathbf{a}) = \arg\max_{\mathbf{w}} \frac{P(\mathbf{w})P(\mathbf{a}|\mathbf{w})}{P(\mathbf{a})}.$$

The crux of this problem is that with a vocabulary size V and utterance length N, the number of possible word-sequences is $\mathrm{O}(V^N)$, i.e., it grows exponentially in the utterance length. Over the years, the process of finding this word sequence has been one of the most studied aspects of speech recognition with numerous techniques and variations developed, [29,69,2]. Interestingly, in recent years, there has been a renaissance of interest in the simplest of these decoding algorithms: the Viterbi procedure. The development of better HMM compilation techniques along with faster computers has made Viterbi applicable to both large vocabulary recognition and constrained tasks, and therefore this section will focus on Viterbi alone.

5.1 The Viterbi Algorithm

The Viterbi algorithm operates on an HMM graph in order to find the best alignment of a sequence of acoustic frames to the states in the graph. For the purposes of this discussion, we will define an HMM in the classical sense as consisting of states with associated acoustic models, and arcs with associated transition costs. A special non-emitting "start state" α and "final state" ω are specified such that all paths start at $t = 0$ in α and end at $t = N + 1$ in ω. Finally, we will associate a string label (possibly "epsilon" or *null*) with each arc. The semantics of Viterbi decoding can then be very simply stated: the single best alignment of the frames to the states is identified, and the word labels encountered on the arcs of this path are output. Note that in the "straight" HMM framework there is no longer any distinction between acoustic model costs, language model costs, or any other costs. All costs associated with all sources of information must be incorporated in the transition and emission costs that define the network: $b_j(o_t)$ and A_{ij}.

A more precise statement of Viterbi decoding is to find the optimal state sequence $\mathbf{S}^* = s_1, s_2, \ldots, s_N$:

$$\mathbf{S}^* = \arg\max_{\mathbf{S}} \prod_{t=1,\ldots,n} b_{s_t}(o_t) a_{s_t s_{t-1}}.$$

Remarkably, due to the limited-history property of HMMs, this can be done with an extremely simple algorithm [54,73]. We define

(1) $\delta_t(j)$: the cost of the best path ending in state j at time t,
(2) $\Psi_t(j)$: the state preceding state j on the best path ending in state t at time t,
(3) pred(s): the set of states that are s's immediate predecessors in the HMM graph.

These quantities can then be computed for all states and all times according to the recursions

(1) *Initialize*

- $\delta_0(\alpha) = 1$,
- $\Psi_0(s) =$ undefined $\forall s$,
- $\delta_0(s) = 0\ \forall s \neq \alpha$;

(2) *Recursion*

- $\delta_t(s) = \max_{j \in \text{pred}(s)} \delta_{t-1}(j) A_{js} b_t(s)$,
- $\Psi_t(s) = \arg\max_{j \in \text{pred}(s)} \delta_{t-1}(j) A_{js} b_t(s)$.

Thus, to perform decoding, one computes the δs and their backpointers Ψ, and then follows the backpointers backwards from the final state ω at time $N+1$. This produces the best path, from which the arc labels can be read off.

In practice, there are a several issues that must be addressed. The simplest of these is that the products of probabilities that define the δs will quickly underflow arithmetic precision. This can be easily dealt with by representing numbers with their logarithms instead. A more difficult issue occurs when non-emitting states are present throughout the graph. The semantics of null states in this case are that spontaneous transitions are allowed without consuming any acoustic frames. The update for a given time frame must then proceed in two stages:

(1) The δs for emitting states are computed in any order by looking at their predecessors.
(2) The δs for null states are computed by iterating over them in topological order and looking at their predecessors.

The final practical issue is that in large systems, it may be advantageous to use pruning to limit the number of states that are examined at each time frame. In this case, one can maintain a fixed number of "live" states at each time frame. The decoding must then be modified to "push" the δs of the live states at time t to the *successor* states at time $t+1$.

An examination of the Viterbi recursions reveals that for an HMM with A arcs and an utterance of N frames, the runtime is $O(NA)$ and the space required is $O(NS)$. However, it is interesting to note that through the use of a divide-and-conquer recursion, the space used can be reduced to $O(Sk \log_k N)$ at the expense of a runtime of $O(NA \log_k N)$ [98]. This is often useful for processing long conversations, messages or broadcasts. The Viterbi algorithm can be applied to any HMM, and the primary distinction is whether the HMM is explicitly represented and stored in advance, or whether it is constructed "on-the-fly." The following two sections address these approaches.

5.1.1 Statically Compiled Decoding Graphs (HMMs)

Section 4.2.3 illustrated the conversion of an N-gram based language model into a statically compiled HMM, and in terms of decoding efficiency, this is probably the best possible strategy [60,78]. In this case, a large number of optimizations can be applied to the decoding graph [60] at "compile time" so that a highly efficient representation is available at decoding time without further processing. Further, it provides a unified way of treating both large and small vocabulary recognition tasks.

5.1.2 Dynamically Compiled Decoding Graphs (HMMs)

Unfortunately, under some circumstances it is difficult or impossible to statically represent the search space. For example, in a cache-LM [48,49] one increases the probability of recently spoken words. Since it is impossible to know what will be said at compile-time, this is poorly suited to static compilation. Another example is the use of *trigger-LMs* [75] in which the co-occurrences of words appearing throughout a sentence are used to determine its probability; in this case, the use of a long-range word-history makes graph compilation difficult. Or in a dialog application, one may want to create a grammar that is specialized to information that a user has just provided; obviously, this cannot be anticipated at compile time. Therefore, despite its renaissance, the use of static decoding graphs is unlikely to become ubiquitous.

In the cases where dynamic graph compilation is necessary, however, the principles of Viterbi decoding can still be used. Recall that when pruning is used, the δ quantities are pushed forward to their successors in the graph. Essentially what is required for dynamic expansion is to associate enough information with each δ that its set of successor states can be computed on demand. This can be done in many ways, a good example being the strategy presented in [69].

5.2 Multipass Lattice Decoding

Under some circumstances, it is desirable to generate not just a single word hypothesis, but a set of hypotheses, all of which have some reasonable likelihood. There are a number of ways of doing this [69,90,65,71,98], and all result in a compact representation of a set of hypotheses as illustrated in Fig. 7. The states in a word lattice are annotated with time information, and the arcs with word labels. Additionally, the arcs may have the acoustic and language model scores associated with the word occurrence (note that with an n-gram LM, this implies that all paths of length $n-1$

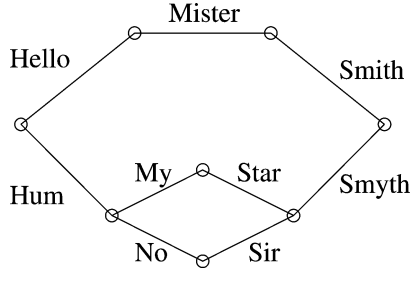

FIG. 7. A word lattice. Any path from the leftmost start state to the rightmost final state represents a possible word sequence.

leading into a state must be labeled with the same word sequence). We note also, that the posterior probability of a word occurrence in a lattice can be computed as the ratio of the sum likelihood of all the paths through the lattice that use the lattice link, to the sum likelihood of all paths entirely. These quantities can be computed with recursions analogous to the HMM $\alpha\beta$ recursions, e.g., as in [98].

Once generated, lattices can be used in a variety of ways. Generally, these involve recomputing the acoustic and language model scores in the lattice with more sophisticated models, and then finding the best path with respect to these updated scores. Some specific examples are:

- Lattices are generated with an acoustic model in which there is no cross-word acoustic context, and then rescored with a model using cross-word acoustic context, e.g., [58,46].
- Lattices are generated with a speaker-independent system, and then rescored using speaker-adapted acoustic models, e.g., [93].
- Lattices are generated with a bigram LM and then rescored with a trigram or 4-gram LM, e.g., [93,55].

The main potential advantage of using lattices is that the rescoring operations can be faster than decoding from scratch with sophisticated models. With efficient Viterbi implementations on static decoding graphs, however, it is not clear that this is the case [78].

5.3 Consensus Decoding

Recall that the decoding procedures that we have discussed so far have aimed at recovering the MAP word hypothesis:

$$\mathbf{w}^* = \arg\max_{\mathbf{w}} P(\mathbf{w}|\mathbf{a}) = \arg\max_{\mathbf{w}} \frac{P(\mathbf{w})P(\mathbf{a}|\mathbf{w})}{P(\mathbf{a})}.$$

Unfortunately, this is not identical to minimizing the WER metric by which speech recognizers are scored. The MAP hypothesis will asymptotically minimize *sentence* error rate, but not necessarily word error rate. Recent work [81,57] has proposed that the correct objective function is really the expected word-error rate under the posterior probability distribution. Denoting the reference or true word sequence by $\mathbf{r}$ and the string edit distance between $\mathbf{w}$ and $\mathbf{r}$ by $E(\mathbf{w}, \mathbf{r})$, the expected error is:

$$E_{P(\mathbf{r}|\mathbf{a})}\big[E(\mathbf{w}, \mathbf{r})\big] = \sum_{\mathbf{r}} P(\mathbf{r}|\mathbf{a})E(\mathbf{w}, \mathbf{r}).$$

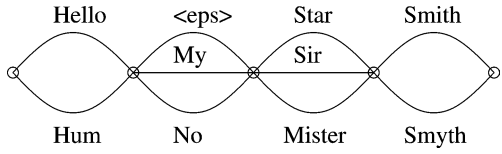

FIG. 8. A word lattice.

Thus, the objective becomes finding

$$\mathbf{w}^* = \arg\max_{\mathbf{w}} \sum_{\mathbf{r}} P(\mathbf{r}|\mathbf{a}) E(\mathbf{w}, \mathbf{r}).$$

There is no known dynamic programming procedure for finding this optimum when the potential word sequences are represented with a general lattice. Therefore, [57] proposes instead work with a segmental or sausage-like structure as illustrated in Fig. 8. To obtain this structure, the links in a lattice are clustered so that temporally overlapping and phonetically similar word occurrences are grouped together. Often, multiple occurrences of the same word (differing in time-alignment or linguistic history) end up together in the same bin, where their posterior probabilities are added together. Under the assumption of a sausage structure, the expected error can then be minimized simply by selecting the link with highest posterior probability in each bin [57]. This procedure has been widely adopted and generally provides a 5 to 10% relative improvement in large vocabulary recognition performance.

5.4 System Combination

In recent DARPA-sponsored speech recognition competitions, it has become common practice to improve the word error rate by combining the outputs of multiple systems. This technique was first developed in [21] where the outputs of multiple systems are aligned to one another, and a voting process is used to select the final output. This process bears a strong similarity to the consensus decoding technique, in that a segmental structure is imposed on the outputs, but differs in its use of multiple systems.

Although the problem of producing an optimal multiple alignment is NP complete [34,21] presents a practical algorithm for computing a reasonable approximation. The algorithm works by iteratively merging a sausage structure that represents the current multiple alignment with a linear word hypothesis. In this algorithm, the system outputs are ordered, and then sequentially merged into a sausage structure.

In a typical use [46], multiple systems are built differing in the front-end analysis, type of training (ML vs. MMI) and/or speaker adaptation techniques that are

used. The combination of 3 to 5 systems may produce on the order of 10% relative improvement over the best single system.

6. Adaptation

The goal of speaker adaptation is to modify the acoustic and language models in light of the data obtained from a specific speaker, so that the models are more closely tuned to the individual. This field has increased in importance since the early 1990s, has been intensively studied, and is still the focus of a significant amount of research. However, since no consensus has emerged on the use of language model adaptation, and many state-of-the-art systems do not use it, this section will focus solely on acoustic model adaptation. In this area, there are three main techniques:

- Maximum A Posteriori (MAP) adaptation, which is the simplest form of acoustic adaptation;
- Vocal Tract Length Normalization (VTLN), which warps the frequency scale to compensate for vocal tract differences;
- Maximum Likelihood Linear Regression, which adjusts the Gaussians and/or feature vectors so as to increase the data likelihood according to an initial transcription.

These methods will be discussed in the following sections.

6.1 MAP Adaptation

MAP adaptation is a Bayesian technique applicable when one has some reasonable expectation as to what appropriate parameter values should be. This prior $g(\theta)$ on the parameters θ is then combined with the likelihood function $f(\mathbf{x}|\theta)$ to obtain the MAP parameter estimates:

$$\theta^* = \arg\max_{\theta} g(\theta) f(\mathbf{x}|\theta).$$

The principled use of MAP estimation has been thoroughly investigated in [28], which presents the formulation that appears here.

The most convenient representation of the prior parameters for p-dimensional Gaussian mixture models is given by Dirichlet priors for the mixture weights $w_1, \ldots, w_K$, and normal-Wishart densities for the Gaussians (parameterized by means m_i and inverse covariance matrices r_i). These priors are expressed in terms of the following parameters:

- ν_k; a count $\nu_k > 0$,
- τ_k; a count $\tau_k > 0$,
- α_k; a count $\alpha_k > p - 1$,
- μ_k; a p dimensional vector,
- u_k; a $p \times p$ positive definite matrix.

Other necessary notation is:

- c_{kt}: the posterior probability of Gaussian k at time t,
- K: the number of Gaussians,
- n: the number of frames.

With this notation, the MAP estimate of the Gaussian mixture parameters are:

$$w'_k = \frac{\nu_k - 1 + \sum_{t=1}^{n} c_{kt}}{n - K + \sum_{k=1}^{K} \nu_k}, \quad m'_k = \frac{\tau_k \mu_k + \sum_{t=1}^{n} c_{kt} x_t}{\tau_k + \sum_{t=1}^{n} c_{kt}},$$

$$r_k'^{-1} = \frac{u_k + \tau_k (\mu_k - m'_k)(\mu_k = m'_k)^{\mathrm{T}}}{\alpha_k - p + \sum_{t=1}^{n} c_{kt}} + \frac{\sum_{t=1}^{n} c_{kt} (x_t - m'_k)(x_t = m'_k)^{\mathrm{T}}}{\alpha_k - p + \sum_{t=1}^{n} c_{kt}}.$$

Unfortunately, there are a large number of free parameters in the representation of the prior, making this formulation somewhat cumbersome in practice. [28] discusses setting these, but in practice it is often easier to work in terms of fictitious counts. Recall that in EM, the Gaussian parameters are estimated from first and second-order sufficient statistics accumulated over the data. One way of obtaining reasonable priors is simply to compute these over the entire training set without regard to phonetic state, and then to weight them according to the amount of emphasis that is desired for the prior. Similarly, statistics computed for one corpus can be downweighted and added to the statistics from another.

6.2 Vocal Tract Length Normalization

The method of VTLN is motivated by the fact that formants and spectral power distributions vary in a systematic way from speaker to speaker. In part, this can be viewed as a side-effect of a speech generation model in which the vocal tract can be viewed as a simple resonance tube, closed at one end. In this case the first resonant frequency is given by $1/L$, where L is the vocal tract length. While such a model is too crude to be of practical use, it does indicate a qualitative relationship between vocal tract length and formant frequencies. The idea of adjusting for this on a speaker-by-speaker basis is old, dating at least to the 1970s [85,7], but was revitalized by a CAIP workshop [44], and improved to a fairly standard form in [87]. The

basic idea is to warp the frequency scale so that the acoustic vectors of a speaker are made more similar to a canonical speaker-independent model. (This idea of "canonicalizing" the feature vectors will recur in another form in Section 6.3.2.) Figure 9 illustrates the form of one common warping function.

There are a very large number of variations on VTLN, and for illustration we choose the implementation presented in [87]. In this procedure, the FFT vector associated with each frame is warped according a warping function like that in Fig. 9. Ten possible warping scales are considered, ranging in the slope of the initial segment from 0.88 to 1.2. The key to this technique is to build a simple model of voiced speech, consisting of a single mixture of Gaussians trained on frames that are identified as being voiced. (This identification is made on the basis of a cepstral analysis described in [40].) To train the voicing model, each speaker is assigned an initial warp scale of 1, and then the following iterative procedure is used:

(1) Using the current warp scales for each speaker, train a GMM for the voiced frames.
(2) Assign to each speaker the warp scale that maximizes the likelihood of his or her warped features according to the current voicing model.
(3) Go to 1.

After several iterations, the outcome of this procedure is a voicing scale for each speaker, and a voicing model. Histograms of the voicing scales are generally bimodal, with one peak for men, and one for women. Training of the standard HMM parameters can then proceed as usual, using the warped or canonicalized features.

The decoding process in similar. For the data associated with a single speaker, the following procedure is used:

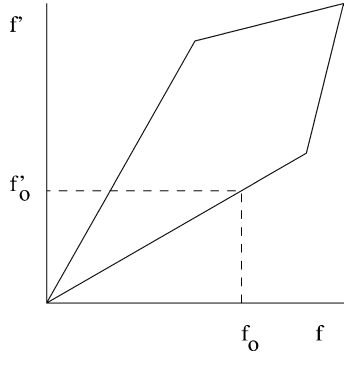

FIG. 9. Two VTLN warping functions. f_0 is mapped into f_0'.

(1) Select the warp scale that maximizes the likelihood of the warped features according to the voicing model.
(2) Warp the features and decode as usual.

The results reported in [87] indicate a 12% relative improvement in performance over unnormalized models, and improvements of this scale are typical [89,96].

As mentioned, a large number of VTLN variants have been explored. [37,61,89] choose warp scales by maximizing the data likelihood with respect to a full-blown HMM model, rather than a single GMM for voiced frames, and experiment with the size of this model. The precise nature of the warping has also been subject to scrutiny; [37] uses a piecewise linear warp with two discontinuities rather than one; [61] experiments with a power law warping function of the form

$$f' = \left(\frac{f}{f_N}\right)^{\beta} f_N$$

where f_N is the bandwidth and [96] experiments with bilinear warping functions of the form

$$f' = f + 2\arctan\left(\frac{(1-\alpha)\sin(f)}{1-(1-\alpha)\cos(f)}\right).$$

Generally, the findings are that piecewise linear models work as well as the more complex models, and that simple acoustic models can be used to estimate the warp factors.

The techniques described so far operate by finding a warp scale using the principles of maximum likelihood estimation. An interesting alternative presented in [20, 32] is based on normalizing formant positions. In [20], a warping function of the form

$$f' = k_s^{3f/8000}$$

is used, where k_s is the ratio of the speaker's third formant to the average frequency of the third formant. In [32], the speaker's first, second, and third formants are plotted against their average values, and the slope of the line fitting these points is used as the warping scale. These approaches, while nicely motivated, have the drawback that it is not easy to identify formant positions, and they have not been extensively adopted.

6.3 MLLR

A seminal paper [52] sparked intensive interest in the mid 1990s in techniques for adapting the means and/or variances of the Gaussians in an HMM model. Whereas VTLN can be thought of as a method for standardizing acoustics across speakers,

Maximum Likelihood Linear Regression was first developed as a mechanism for adapting the acoustic models to the peculiarities of individual speakers. This form of MLLR is known as "model-space" MLLR, and is discussed in the following section. It was soon realized [18,25], however, that one particular form of MLLR has an equivalent interpretation as on operation on the features, or "feature-space" MLLR. This technique is described in Section 6.3.2, and can be thought of as another canonicalizing operation.

6.3.1 Model Space MLLR

A well defined question first posed in [52] is, suppose the means of the Gaussians are transformed according to

$$\hat{\boldsymbol{\mu}} = A\boldsymbol{\mu} + \mathbf{b}.$$

Under the assumption of this form of transform, what matrix A and offset vector $\mathbf{b}$ will maximize the data probability given an initial transcription of the data? To solve this, one defines an extended mean vector

$$\boldsymbol{\xi} = [1\mu_1\mu_2 \ldots \mu_p]^{\mathrm{T}}$$

and a $p \times p+1$ matrix W. The likelihood assigned by a Gaussian g is then given by

$$N(\mathbf{x}; W\boldsymbol{\xi}_g, \Sigma_g).$$

In general, with a limited amount of training data, it may be advantageous to tie the transforms of many Gaussians, for example all those belonging to a single phone or phone-group such as vowels. If we define $\gamma_{g(t)}$ to be the posterior probability of Gaussian g having generated the observation o_t at time t, and $\mathcal{G}$ to be the set of Gaussians whose transforms are to be tied, then the matrix W is given by the following equation [52]

$$\sum_{t=1}^{t=N} \sum_{g \in \mathcal{G}} \gamma_g(t) \Sigma_g^{-1} \mathbf{o}(t) \boldsymbol{\xi}_g^{\mathrm{T}} = \sum_{t=1}^{t=N} \sum_{g \in \mathcal{G}} \gamma_g(t) \Sigma_g^{-1} W \boldsymbol{\xi}_g \boldsymbol{\xi}_g^{\mathrm{T}}.$$

Thus, estimating the transforms simply requires accumulating the sufficient statistics used in ML reestimation, and solving a simple matrix equation. Choosing the sets of Gaussians to tie can be done simply by clustering the Gaussians according to pre-defined phonetic criteria, or according to KL divergence [53]. Depending on the amount of adaptation data available, anywhere from 1 to several hundred transforms may be used.

A natural extension of mean-adaptation is to apply a linear transformation to the Gaussian variances as well [25,23]. The form of this transformation is given by

$$\hat{\boldsymbol{\mu}} = A\boldsymbol{\mu} + \mathbf{b} = W\boldsymbol{\xi}$$

and

$$\widehat{\Sigma} = H \Sigma H^{\mathrm{T}}$$

where W and H are the matrices to be estimated. A procedure for doing this is presented in [23].

6.3.2 Feature Space MLLR

Although it is a constrained version of the mean and variance transform described in the previous section, in some ways the most important form of MLLR applies the same transform to the means as to the variances:

$$\hat{\boldsymbol{\mu}} = A' \boldsymbol{\mu} - \mathbf{b}', \quad \widehat{\Sigma} = A' \Sigma A'^{\mathrm{T}}.$$

Under this constraint, straightforward estimation formulae can be derived, but more importantly, the transformation can be applied in to the feature vectors rather than the models, according to:

$$\hat{\mathbf{o}}(t) = A'^{-1} \mathbf{o}(t) + A'^{-1} \mathbf{b}' = A \mathbf{o}(t) + \mathbf{b}.$$

The likelihoods computed with this feature transformation differ from those computed with the model transform by $\log(|A|)$. When, as is often done, a single fMLLR transform is used, this can be ignored in Viterbi decoding and EM training. This has two important ramifications. Once the transforms have been estimated,

(1) Transformed features can be written out and the models can be retrained with the standard EM procedures (speaker-adaptive or SAT training) and
(2) MMI or other discriminative training can be performed with the transformed features.

Curiously, although multiple MLLR transforms are commonly used, the use of multiple fMLLR transforms has not yet been thoroughly explored. Due to the convenience of working with transformed or canonicalized features, feature space MLLR has become a common part of modern systems [78,93]. It is often used in conjunction with VTLN in the following speaker-adaptive or SAT training procedure:

(1) Train a speaker-independent (SI) system.
(2) Estimate VTLN warp scales using the frames that align to voiced phones with the SI system.
(3) Write out warped features for each speaker.
(4) Train a VTLN-adapted system.
(5) Estimate fMLLR transforms with the VTLN models.
(6) Write out fMLLR-VTLN features.
(7) Train ML and/or MMI systems from the canonical features.

7. Performance Levels

In order to illustrate the error rates attainable with today's technology—and the relative contribution of the techniques discussed in earlier sections—the following paragraphs describe the state-of-the-art as embodied by an advanced IBM system in 2002 [46]. This system was designed to work well across a wide variety of speakers and topics, and is tested on five separate datasets:

(1) Telephone conversations (Swb98).
(2) Meeting recordings (mtg).
(3) Two sets of call center recordings of customers discussing account information (cc1 and cc2).
(4) Voicemail recordings (vm).

In this system, the recognition steps are as follows:

P1 Speaker-independent decoding. The system uses mean-normalized MFCC features and an acoustic model with 4078 left context-dependent states and 171K mixture components.
P2 VTLN decoding. VTLN warp factors are estimated for each speaker using forced alignments of the data to the recognition hypotheses from P1, then recognition is performed with a VTLN system that uses mean-normalized PLP features and an acoustic model with 4440 left context-dependent states and 163K mixture components.
P3 Lattice generation. Initial word lattices are generated with a SAT system that uses mean-normalized PLP features and an acoustic model with 3688 word-internal context-dependent states and 151K mixture components. FMLLR transforms are computed with the recognition hypotheses from P2.
P4 Acoustic rescoring with large SAT models. The lattices from P3 are rescored with five different SAT acoustic models and pruned. The acoustic models are as follows:
   A An MMI trained PLP system with 10437 left context-dependent states and 623K mixture components. The maximum value of $c0$ is subtracted from each feature vector, and mean-normalization is performed for the other cepstral coefficients.
   B An MLE PLP system identical to the system of P4A, except for the use of MLE training of the acoustic model.
   C An MLE PLP system with 10450 left context-dependent states and 589K mixture components. This system uses mean normalization of all raw features including $c0$.
   D A SPAM MFCC system with 10133 left context-dependent states and 217K mixture components.

E An MLE MFCC system with 10441 left context-dependent states and 600K mixture components. This system uses max.-normalization of $c0$ and mean normalization of all other raw features.

The FMLLR transforms for each of the five acoustic models are computed from the one-best hypotheses in the lattices from P3.

P5 Acoustic model adaptation. Each of the five acoustic models are adapted with MLLR using one-best hypotheses from their respective lattices generated in P4.

P6 4-gram rescoring. Each of the five sets of lattices from P5 are rescored and pruned using a 4-gram language model.

P7 Confusion network combination. Each of the five sets of lattices from P6 are processed to generate confusion networks [57], then a final recognition hypothesis is generated by combining the confusion networks for each utterance.

The performance of the various recognition passes on the test set is summarized in Table I.

TABLE I
WORD ERROR RATES (%) FOR EACH TEST SET AT EACH PROCESSING STAGE AND THE OVERALL, AVERAGE ERROR RATE. FOR PASSES WHERE MULTIPLE SYSTEMS ARE USED (P4–6), THE BEST ERROR RATE FOR A TEST COMPONENT IS HIGHLIGHTED

Pass	swb98	mtg	cc1	cc2	vm	All
P1	42.5	62.2	67.8	47.6	35.4	51.1
P2	38.7	53.7	56.9	44.1	31.7	45.0
P3	36.0	44.6	46.6	40.1	28.0	39.1
P4A	**31.5**	**39.4**	41.7	38.2	26.7	35.5
P4B	32.3	40.0	**41.3**	39.0	26.7	35.9
P4C	32.5	40.2	42.1	39.9	27.0	36.3
P4D	31.7	40.3	42.6	**37.6**	**25.8**	35.6
P4E	33.0	40.5	43.4	38.8	26.9	36.5
P5A	30.9	**38.3**	**39.4**	36.9	26.1	34.3
P5B	31.5	38.5	**39.4**	37.0	26.5	34.6
P5C	31.6	38.7	41.0	39.4	26.8	35.5
P5D	**30.8**	39.0	41.1	**36.7**	**25.6**	34.6
P5E	32.1	38.9	41.8	36.8	26.4	35.2
P6A	**30.4**	**38.0**	**38.9**	36.5	25.7	33.9
P6B	31.0	38.3	**38.9**	36.4	25.8	34.1
P6C	31.2	38.4	40.1	38.9	26.3	35.0
P6D	**30.4**	38.6	40.8	36.3	**25.5**	34.3
P6E	31.5	38.5	41.6	**35.9**	25.7	34.6
P7	29.0	35.0	37.9	33.6	24.5	32.0

8. Conclusion

Over the past decade, incremental advances in HMM technology have advanced the state of the art to the point where commercial use is possible. These advances have occurred in all areas of speech recognition, and include

- LDA and HLDA analysis in feature extraction,
- discriminative training,
- VTLN, MLLR and FMLLR for speaker adaptation,
- the use of determinization and minimization in decoding graph compilation,
- consensus decoding,
- voting and system combination.

Collectively applied, these advances produce impressive results for many speakers under many conditions. However, under some conditions, such as when background noise is present or speech is transmitted over a low-quality cell phone or a speaker has an unusual accent, today's systems can fail. As the error-rates of Section 7 illustrate, this happens enough that the average error-rate for numerous tasks across a variety of conditions is around 30%—far from human levels. Thus, the most critical problem over the coming decade is develop truly robust techniques that reduce the error rate by another factor of five.

References

[1] Aho A.V., Sethi R., Ullman J.D., *Compilers: Principles, Techniques, and Tools*, Addison–Wesley, Reading, MA, 1986.
[2] Aubert X., "A brief overview of decoding techniques for large vocabulary continuous speech recognition", in: *Automatic Speech Recognition: Challenges for the New Millennium*, 2000.
[3] Axelrod S., Gopinath R., Olsen P., "Modeling with a subspace constraint on inverse covariance matrices", in: *ICSLP*, 2002.
[4] Bahl L.R., Brown P.F., de Souza P.V., Mercer R.L., "Maximum mutual information estimation of hidden Markov model parameters for speech recognition", in: *ICASSP*, 1986, pp. 49–52.
[5] Bahl L.R., et al., "Context dependent modeling of phones in continuous speech using decision trees", in: *Proceedings of DARPA Speech and Natural Language Processing Workshop*, 1991.
[6] Baker J., "The Dragon system—an overview", *IEEE Transactions on Acoustics, Speech, and Signal Processing* **23** (1975) 24–29.
[7] Bamberg P., "Vocal tract normalization", Technical report, Verbex, 1981.

[8] Brown P.F., et al., "Class-based *n*-gram models of natural language", *Comput. Linguist.* **18** (1992).
[9] Chen S., Eide E., Gales M., Gopinath R., Olsen P., "Recent improvements in IBM's speech recognition system for automatic transcription of broadcast speech", in: *Proceedings of the DARPA Broadcast News Workshop*, 1999.
[10] Chen S.S., Gopalakrishnan P.S., "Clustering via the Bayesian information criterion with applications in speech recognition", in: *ICASSP*, 1995, pp. 645–648.
[11] Chen S., et al., "Speech recognition for DARPA communicator", in: *ICASSP*, 2001.
[12] Chen S.F., Goodman J., "An empirical study of smoothing techniques for language modeling", Technical Report TR-10-98, Harvard University, 1998.
[13] Chomsky N., *Aspects of the Theory of Syntax*, MIT Press, Cambridge, MA, 1965.
[14] CMU, *The CMU Pronouncing Dictionary*, 2003.
[15] Linguistic Data Consortium, *Callhome American English lexicon (pronlex)*, 2003.
[16] Davies K., et al., "The IBM conversational telephony system for financial applications", in: *Eurospeech*, 1999.
[17] Davis S., Mermelstein P., "Comparison of parametric representations for monosyllabic word recognition in continuously spoken sentences", *IEEE Transactions on Acoustics, Speech, and Signal Processing* **28** (1980) 357–366.
[18] Digalakis V.V., Rtischev D., Neumeyer L.G., "Speaker adaptation using constrained estimation of Gaussian mixtures", *IEEE Transactions on Speech and Audio Processing* (1995) 357–366.
[19] Duda R.O., Hart P.B., *Pattern Classification and Scene Analysis*, Wiley, New York, 1973.
[20] Eide E., Gish H., "A parametric approach to vocal tract length normalization", in: *ICASSP*, 1996, pp. 346–348.
[21] Fiscus J.G., "A post-processing system to yield reduced word error rates: Recognizer output voting error reduction (rover)", in: *IEEE Workshop on Automatic Speech Recognition and Understanding*, 1997.
[22] Furui S., "Speaker independent isolated word recognition using dynamic features of speech spectrum", *IEEE Transactions on Acoustics Speech and Signal Processing* **34** (1986) 52–59.
[23] Gales M.J.F., "Maximum likelihood linear transformations for HMM-based speech recognition", Technical Report CUED-TR-291, Cambridge University, 1997.
[24] Gales M.J.F., "Maximum likelihood linear transformations for HMM-based speech recognition", *Computer Speech and Language* **12** (1998).
[25] Gales M.J.F., Woodland P.C., "Mean and variance adaptation within the MLLR framework", *Computer Speech and Language* **10** (1996) 249–264.
[26] Gao Y., Ramabhadran B., Chen J., Erdogan H., Picheny M., "Innovative approaches for large vocabulary name recognition", in: *ICASSP*, 2001.
[27] Gauvain J.-L., Lamel L., Adda G., "The LIMSI 1999 BN transcription system", in: *Proceedings 2000 Speech Transcription Workshop*, 2000, http://www.nist.gov/speech/publications/tw00/html/abstract.htm.

[28] Gauvain J.-L., Lee C.-H., "Maximum a posteriori estimation for multivariate Gaussian mixture observations of Markov chains", *IEEE Transactions on Speech and Audio Processing* **2** (1994) 291–298.
[29] Gopalakrishnan P.S., Bahl L.R., Mercer R.L., "A tree-search strategy for large vocabulary continuous speech recognition", in: *ICASSP*, 1995.
[30] Gopalakrishnan P., Kanevsky D., Nadas A., Nahamoo D., "An inequality for rational functions with applications to some statistical estimation problems", *IEEE Transactions on Information Theory* **37** (1991) 107–113.
[31] Gopinath R., "Maximum likelihood modeling with Gaussian distributions for classification", in: *ICASSP*, 1998.
[32] Gouvea E.B., Stern R.M., "Speaker normalization through formant-based warping of the frequency scale", in: *Eurospeech*, 1997.
[33] Graff D., *The English Gigaword Corpus*, 2003.
[34] Gusfield D., *Algorithms on Strings, Trees and Sequences*, Cambridge Univ. Press, Cambridge, UK, 1997.
[35] Haeb-Umbach R., Ney H., "Linear discriminant analysis for improved large vocabulary continuous speech recognition", in: *ICASSP*, 1992.
[36] Hain T., Woodland P.C., Evermann G., Povey D., "The CU-HTK March 2000 HUB5E transcription system", in: *Proc. Speech Transcription Workshop*, 2000.
[37] Hain T., Woodland P.C., Niesler T.R., Whittaker E.W.D., "The 1998 HTK system for transcription of conversational telephone speech", in: *Eurospeech*, 1999.
[38] Hermansky H., "Perceptual linear predictive (PLP) analysis of speech", *J. Acoustical Society of America* **87** (1990) 1738–1752.
[39] Hopcroft J.E., Ullman J.D., *Introduction to Automata Theory, Languages and Computation*, Addison–Wesley, Reading, MA, 1979.
[40] Hunt M.J., "A robust method of detecting the presence of voiced speech", in: *ICASSP*, 1995.
[41] Jan E., Maison B., Mangu L., Zweig G., "Automatic construction of unique signatures and confusable sets for natural language directory assistance applications", in: *Eurospeech*, 2003.
[42] Jeffreys H., *Theory of Probability*, Clarendon, Oxford, 1948.
[43] Jelinek F., "Continuous speech recognition by statistical methods", *Proceedings of the IEEE* **64** (1976) 532–556.
[44] Kamm T., Andreou A., Cohen J., "Vocal tract normalization in speech recognition: Compensating for systematic speaker variability", in: *Proceedings of the 15th Annual Speech Recognition Symposium, Baltimore, MD*, 1995, pp. 175–178.
[45] Katz S.M., "Estimation of probabilities from sparse data for the language model component of a speech recognizer", *IEEE Transactions of Acoustics, Speech and Signal Processing* **35** (1987) 400–401.
[46] Kingsbury B., Mangu L., Saon G., Zweig G., Axelrod S., Visweswariah K., Picheny M., "Towards domain independent conversational speech recognition", in: *Eurospeech*, 2003.
[47] Kneser N., Ney H., "Improved backing-off for m-gram language modeling", in: *ICASSP*, 1995.

[48] Kuhn R., "Speech recognition and the frequency of recently used words: A modified Markov model for natural language", in: *12th International Conference on Computational Linguistics, Budapest*, 1988, pp. 348–350.
[49] Kuhn R., De Mori R., "A cache based natural language model for speech recognition", *IEEE Transactions on Pattern Analysis and Machine Intelligence* **12** (1990) 570–583.
[50] Kumar N., Andreou A.G., "Heteroscedastic discriminant analysis and reduced rank HMMs for improved speech recognition", *Speech Communication* (1998) 283–297.
[51] Leggetter C., Woodland P.C., "Flexible speaker adaptation using maximum likelihood linear regression", in: *Eurospeech*, 1995.
[52] Leggetter C., Woodland P.C., "Speaker adaptation of continuous density HMMs using multivariate linear regression", in: *ICSLP*, 1994.
[53] Leggetter C.J., Woodland P.C., "Flexible speaker adaptation using maximum likelihood linear regression", in: *Eurospeech*, 1995.
[54] Levinson S.E., Rabiner L.R., Sondhi M.M., "An introduction to the application of the theory of probabilistic functions of a Markov process to automatic speech recognition", *The Bell System Technical Journal* **62** (1983) 1035–1074.
[55] Ljolje A., et al., "The AT&T 2000 LVSCR system", in: *Proceedings 2000 Speech Transcription Workshop*, 2000, http://www.nist.gov/speech/publications/tw00/html/abstract.htm.
[56] Makhoul J., "Linear prediction: A tutorial review", *Proceedings of the IEEE* **63** (1975) 561–580.
[57] Mangu L., Brill E., Stolcke A., "Finding consensus in speech recognition: Word error minimization and other applications of confusion networks", *Computer Speech and Language* **14** (2000) 373–400.
[58] Matsoukas S., et al., "Speech to text research at BBN", in: *Proceedings of January 2003 EARS Midyear Meeting*, 2003.
[59] Mohri M., "Finite-state transducers in language and speech processing", *Comput. Linguist.* **23** (1997).
[60] Mohri M., Riley M., Hindle D., Ljolje A., Pereira F., "Full expansion of context-dependent networks in large vocabulary speech recognition", in: *ICASSP*, 1998.
[61] Molau S., Kanthak S., Ney H., "Efficient vocal tract normalization in automatic speech recognition", in: *ESSV*, 2000, pp. 209–216.
[62] Nadas A., "A decision theoretic formulation of a training problem in speech recognition and a comparison of training by unconditional versus unconditional maximum likelihood", *IEEE Transactions on Acoustics, Speech, and Signal Processing* **31** (1983).
[63] Nadas A., Nahamoo D., Picheny M., "On a model-robust training method for speech recognition", *IEEE Transactions on Acoustics, Speech, and Signal Processing* **36** (1988).
[64] Naur P., "Revised report on the algorithmic language Algol 60", *Communications of the Association for Computing Machinery* **6** (1963) 1–17.
[65] Neukirchen C., Klakow D., Aubert X., "Generation and expansion of word graphs using long span context information", in: *ICASSP*, 2001.
[66] Ney H., Essen U., Kneser R., "On structuring probabilistic dependences in stochastic language modelling", *Computer Speech and Language* (1994) 1–38.

[67] Niesler T.R., Whittaker E.W.D., Woodland P.C., "Comparison of part-of-speech and automatically derived category-based language models for speech recognition", in: *ICASSP*, 1998.

[68] Normandin Y., Regis C., De Mori R., "High-performance connected digit recognition using maximum mutual information", *IEEE Transactions on Speech and Audio Processing* **2** (1994) 299–311.

[69] Odell J.J., "The use of context in large vocabulary speech recognition", Cambridge University dissertation, 1995.

[70] Olsen P., Gopinath R., "Extended MLLT for Gaussian mixture models", *IEEE Transactions on Speech and Audio Processing* (2001).

[71] Ortmanns S., Ney H., "A word graph algorithm for large vocabulary continuous speech recognition", *Computer Speech and Language* (1997) 43–72.

[72] Pellom B., Ward W., Hansen J., Hacioglu K., Zhang J., Yu X., Pradhan S., "University of Colorado dialog systems for travel and navigation", in: *Human Language Technologies*, 2001.

[73] Rabiner L.R., Juang B.-H., "An introduction to hidden Markov models", *IEEE ASSP Magazine* (1986) 4–16.

[74] Rabiner L.R., Juang B.-H., *Fundamentals of Speech Recognition*, Prentice Hall, New York, 1993.

[75] Rosenfeld R., "A maximum entropy approach to adaptive statistical language modeling", *Computer Speech and Language* **10** (1996) 187–228.

[76] Sankar A., Gadde V.R.R., Stolcke A., Weng F., "Improved modeling and efficiency for automatic transcription of broadcast news", *Speech Communication* **37** (2002) 133–158.

[77] Saon G., Padmanabhan M., Gopinath R., Chen S., "Maximum likelihood discriminant feature spaces", in: *ICASSP*, 2000.

[78] Saon G., Zweig G., Kingsbury B., Mangu L., Chaudhari U., "An architecture for rapid decoding of large vocabulary conversational speech", in: *Eurospeech*, 2003.

[79] Schroeder M.R., "Recognition of complex acoustic signals", in: Bullock T.H. (Ed.), *Life Sciences Research Report 5*, Abakon Verlag, 1977.

[80] Seymore K., Rosenfeld R., "Scalable backoff language models", in: *ICSLP*, 1996.

[81] Stolcke A., Konig Y., Weintraub M., "Explicit word error minimization using n-best list rescoring", in: *Eurospeech*, 1997.

[82] Stolcke A., "Entropy-based pruning of backoff language models", in: *Proceedings of DARPA Broadcast News Transcription and Understanding Workshop*, 1998, pp. 270–274.

[83] Stolcke A., "Srilm—an extensible language modeling toolkit", in: *ICSLP*, 2002.

[84] Suontausta J., Hakkinen J., Olli V., "Fast decoding in large vocabulary name dialing", in: *ICASSP*, 2000, pp. 1535–1538.

[85] Waitika H., "Normalization of vowels by vocal-tract length and its application to vowel identification", *IEEE Transactions on Audio Speech and Signal Processing* (1977) 183–192.

[86] Wegmann S., Zhan P., Carp I., Newman M., Yamron J., Gillick L., "Dragon systems' 1998 broadcast news transcription system", in: *Proceedings of the DARPA Broadcast News Workshop*, NIST, 1999.

[87] Wegmann S., McAllaster D., Orloff J., Peskin B., "Speaker normalization on conversational telephone speech", in: *ICASSP*, 1996.

[88] Welling L., Haberland N., Ney H., "Acoustic front-end optimization for large vocabulary speech recognition", in: *Eurospeech*, 1997.

[89] Welling R., Haeb-Umbach R., Aubert X., Haberland N., "A study on speaker normalization using vocal tract normalization and speaker adaptive training", in: *ICASSP*, 1998, pp. 797–800.

[90] Weng F., Stolcke A., Sankar A., "Efficient lattice representation and generation", in: *ICSLP*, 1998.

[91] Whittaker E.W.D., Woodland P.C., "Efficient class-based language modelling for very large vocabularies", in: *ICASSP*, 2001.

[92] Woodland P.C., Povey D., "Large scale discriminative training for speech recognition", in: *Automatic Speech Recognition: Challenges for the New Millennium*, 2000.

[93] Woodland P., et al., "The CU-HTK April 2002 switchboard system", in: *EARS Rich Transcription Workshop*, 2002.

[94] Young S., Odell J., Ollason D., Valtchev V., Woodland P., *The HTK Book*, 2.1 edition, Entropic Cambridge Research Laboratory, 1997.

[95] Young S.J., Odell J.J., Woodland P.C., "Tree-based tying for high accuracy acoustic modelling", in: *ARPA Workshop on Human Language Technology*, 1994.

[96] Zhan P., Waibel A., "Vocal tract length normalization for large vocabulary continuous speech recognition", Technical Report CMU-CS-97-148, School of Computer Science, Carnegie Mellon University, 1997.

[97] Zue V., et al., "A telephone-based conversational interface for weather information", 2000.

[98] Zweig G., Padmanabhan M., "Exact alpha–beta computation in logarithmic space with application to map word graph construction", in: *ICSLP*, 2000.

[99] Zweig G., Saon G., Yvon F., "Arc minimization in finite state decoding graphs with cross-word acoustic context", in: *ICSLP*, 2002.

[100] Zwicker E., "Subdivision of the audible frequency range into critical bands", *J. Acoustical Society of America* **33** (1961) 248.

[101] Zwicker E., "Masking and physiological excitation as consequences of ear's frequency analysis", in: Plomp R., Smoorenburg G.F. (Eds.), *Frequency Analysis and Periodicity Detection in Hearing*, 1970.