Internal Use Only

002

TR-IT-0147

Speech Recognition System Design using Automatically Learned Non-Uniform Segmental Units

Mari Ostendorf Michiel Bacchiani Y. Sagisaka

K. Paliwal

1996.01

This report describes a new approach to acoustic modeling that combines segmental acoustic modeling with automatic learning techniques to determine the appropriate acoustic units. Since the units are not constrained to be phonetic (or tied to any other linguistic unit) and can optionally span word boundaries, an important problem is automatic determination of the lexicon, which includes both the list of lexical entries and the "pronunciation" in terms of the segmental units. The solution to these problems is outlined below, and some preliminary experimental results are described.

1 Introduction

Speech recognition researchers have long sought methods for automatically learning subword units and representations of words in terms of these units, in order to more efficiently represent words for large vocabulary speech recognition and take best advantage of limited training data. Automatic learning of units for speech recognition was an active research topic in the late 1980's [1, 2, 3], with the motivation being that human labor required to build phonetic baseforms was costly and error-prone and that recognition performance could be improved by using more detailed units than phones. However, research in automatic learning of sub-word units lost momentum in the early 90's when large dictionaries became more widely available and context-dependent phone modeling with distribution clustering techniques [4, 5, 6, 7, 8, 9] allowed for efficient acoustic sub-word modeling without the added problem of finding a lexical mapping. Research in lexical mapping proceeded in two new directions then, focusing on problems of finding pronunciations for out-of-vocabulary words (particularly important for names) and designing phonetic pronunciation networks to allow for some variability in word pronunciation.

While great progress has been made in these areas for recognition of continuous read speech, recognition of conversational speech still lags far behind in performance. For example, the same basic system that achieves 90% recognition accuracy on unlimited vocabulary read Wall Street Journal speech and 97% accuracy on a roughly 5000 word vocabulary spontaneous human-computer database query task [10], achieves roughly 50% accuracy on spontaneous human-human dialog utterances in benchmarks on the Switchboard corpus. This suggests that a radical shift in modeling is needed to handle some of the phenomena in spontaneous speech, and that it is worthwhile revisiting the problem of automatic learning of acoustic units.

Two problems are faced in automatic learning of acoustic units:

- searching for acoustic units that optimize some objective function, e.g. maximum likelihood, and
- finding a mapping from these units to the lexicon.

Here, we will address the first problem by following the same basic approach as outlined in early work, but extending it to accommodate segmental models with parametric mean trajectories and to use iterative maximum likelihood estimation. In addition, an important difference in our work is that we allow non-uniform unit size: units may be bigger or smaller than a phone and may span word boundaries. Because we allow units that span word boundaries, the problem of finding a mapping from these units to the lexicon becomes a two step problem of first finding the lexicon (since it may sometimes be appropriate to include word strings in addition to single words) and then finding the mapping from the segmental units to the lexicon. The approach to lexical mapping will involve estimating a probabilistic network, using a hybrid of previously proposed approaches for lexical modeling.

The remainder of the report is structured as follows. Section 2 describes the approach to learning recognition units, which builds on work in speech coding and segment recognition. Next, Section 3 outlines the approach to finding the lexicon as well as the mapping from the segmental units to lexical entries, i.e. the word "pronunciations". In Section 4, we outline different issues associated with the speech recognition search algorithm, which is more difficult with a segmental model. Preliminary experimental results are reported in Section 5, followed by a summary of the work and discussion of possible extensions in Section 6.

2 Unsupervised Learning of a Segmental Inventory

The first step in acoustic modeling with non-uniform units is clearly finding the units. In this section, we summarize previous work on this problem based on hidden Markov modeling and speech coding, as well as previous work on segmental modeling of phonetic units, and then show how the two techniques can be combined to automatically derive segmental units for speech recognition.

2.1 Previous Work

In work aimed at automatic segmentation of speech, Svendsen and Soong [11] describe an algorithm for finding quasi-stationary regions of speech without phone transcriptions, by using a dynamic programming algorithm to find the maximum likelihood segmentation, assuming a multivariate Gaussian model for each segment with a single, diagonal covariance used for all segments. In their approach, the desired number of segments in an utterance was pre-specified. A variation of this algorithm was used by Lee *et al.* [1] to find a segmentation assuming an autoregressive stochastic process model of speech and using the Itakura-Saito distortion measure. Lee *et al.* took the work further, however, by applying it to the problem of learning sub-word units for speech recognition. The resulting segments are then clustered according to the IS distortion and each cluster is then trained as an HMM state. Paliwal [3] used a similar approach in his lexicon-building study, except that he clustered the segment centroids, rather than clustering the data directly as in [1].

At the same time, IBM was developing a technique for representing non-phonetic sub-word units called "fenones". They used vector quantization (VQ) as a segmentation technique, then defined initial fenonic baseforms based on the VQ output sequences associated with a single training token for each word, and finally determined a new baseform using maximum likelihood fenone decoding on a collection of training tokens [2]. The result was a linear string of fenones to represent the pronunciation of a word, extended later to the multone model [12], which effectively introduced mixture distributions.

Automatic learning of sub-word units has long been studied in speech coding, since it is fundamental to the problem of quantization. Most closely related to the speech recognition work are the variable-length speech segment coding techniques. Roucos *et al.* [13] developed the first "segment" coder, which used fixed-length segment codewords and a variable-to-fixed-length mapping to encode the variable length observations. The segment vocoder later inspired the stochastic segment model for recognition [14]. The segment vocoder design algorithm was a simple extension of the standard VQ design algorithm [15], but had the disadvantage that it did not directly minimize quantization error on the decoded segment. This problem was address in work by Shiraki and Honda [16], who assumed that a segment is represented by a sampled regression function, in order to represent variable-length segments with a fixed number of parameters. They also introduced an iterative segmentation/quantizer design algorithm. More recently, Chou and Lookabaugh [17] developed a different segment coder design algorithm that mapped variable-length segments to codewords of the same length (rather than assume a parametric model of the segment) and built on entropy-constrained vector quantization techniques to obtain a more efficient quantizer.

In the work described in this report, our ultimate goal is to use Gaussian models, since continuous distribution models have proved to outperform discrete distributions in recognition tests. (As evidence, consider that all recognition systems participating in the 1994 ARPA benchmark tests used some form of continuous distribution, whether Gaussian mixture or neural network predictor.) Hence the segmentation approach described in [11, 1, 3] is more appropriate for our work than fenonic baseform derivation. However, iterative re-estimation techniques outlined in the coding work point to a method for improving previous work, though we would work with a maximum likelihood rather than minimum distortion objective function. Between the variable-length coding schemes, our intuition suggests that the Shiraki-Honda parametric segment model [16] is the best choice for representing speech where units are stretched and compressed in time.

2.2 Parametric Trajectory Segmental Models

¿From coding theory, we know that quantizing sequences of vectors makes it possible to achieve lower distortion for a given bit rate than quantizing individual samples. If a segmental coding strategy is more efficient than a frame-based coding strategy, it seems reasonable to expect that a segmental recognition strategy will be more effective than a frame-based (e.g. HMM) recognition strategy, since minimum distortion is related to maximum likelihood in a Gaussian model. In addition, an HMM can be viewed as a special case of the class of segment models, so by working with a more general model we allow automatic learning to determine the appropriate structure. Consequently, this work will focus on segment-based modeling, particularly the version that uses a regression function to describe the mean trajectory, which is essentially a probabilistic variation of the Shiraki-Honda segment [16]. In this section, we give a brief description of segment-based modeling, focusing in particular on the parametric mean trajectory case used here. A general overview of segment modeling can be found in [18].

Let y_t be a d-dimensional observation vector, such as a vector of cepstral coefficients representing a window of speech at time t. A general segment model provides a joint model for a random-length sequence of observations $y_1^l = [y_1, \ldots, y_l]$, a segment, generated by unit a according to the density

$$\mathrm{p}(y_1, \ldots, y_l, l|a) = \mathrm{p}(y_1, \ldots, y_l|l, a)\mathrm{p}(l|a) = b_{a,l}(y_1^l)\mathrm{p}(l|a), \tag{1}$$

where l is the segment length. Letting $\mathcal{L}$ be the set of possible observation lengths (in frames), a segment model for label $a \in \mathcal{A}$ is characterized by 1) a *duration distribution* $\mathrm{p}(l|a)$ that gives the likelihood of segment length $l \in \mathcal{L}$ and thereby the likelihood of a particular segmentation of an utterance, and 2) a *family of output densities* $\{b_{a,l}(y_1^l); l \in \mathcal{L}\}$ that describes observation sequences of different lengths. In addition, a Markov assumption for sequences of a_i is made either implicitly or explicitly, by embedding phone segments in a word pronunciation network or other probabilistic finite-state network.

The segment duration distribution $\{\mathrm{p}(l|a);\ l \in \mathcal{L}\}$ can be either parametric (e.g. Gamma) or non-parametric (relative frequency). In practice, any reasonable assumption works well, since the contribution of the duration model is small relative to the segment observation probability which is in a much higher dimensional space. The family of output densities $\{b_{a,l}(y_1^l);\ l \in \mathcal{L}\}$ represents l-length trajectories in vector space ($y_i \in \Re^d$) with a sequence of distributions that can be thought of as dividing the segment into separate regions in time, where a "region" is similar to an HMM state. A distribution mapping (or, time-warping transformation) associates each frame y_i in the variable length observation y_1^l with one of the model regions. Together the mapping and the region-dependent distributions provide a means of specifying $b_{a,l}(y_1^l)$ for a large range of l with a small number of parameters. A variety of options are possible for the mapping T_l, as outlined in [18]. Here, we will assume a deterministic linear time sampling of a fixed mean trajectory.

Parametric mean trajectory segment models were introduced separately by Gish and Ng [19] (as a segment model) and Deng *et al.* [20] (as a nonstationary-state HMM). In both cases, the observations in a segment are assumed to be conditionally independent given the segment length, and the probability of a segment is therefore the product of the probability of each frame:

$$b_{a,l}(y_1^l) = \prod_{i=1}^{l} \mathrm{p}(y_i|a, r_i) \tag{2}$$

where the distribution used for each frame, indexed by region r_i, is Gaussian: $\mathrm{p}(y_i|a, r_i) \sim N(\mu_i, \Sigma)$. The covariance Σ is assumed to be constant throughout the segment, which reduces training costs. The sequence of means μ_i are parameterized by a polynomial in d-dimensional vector space. Specifically, the sequence of distributions used in computing the likelihood of an l-length segment y_1^l is described by the sequence of means $[\mu_1 \ldots \mu_l] = BZ_l$, where B is a $d \times m$ matrix of coefficients for polynomial order $m-1$ and Z_l is an $m \times l$ time sampling matrix. For $m = 1$, Z_l is a row vector of 1's; for $m = 2$, Z_l has a vector of 1's in the first row and a vector of normalized times $\tilde{t}$ in the second row. As an example, the value of the i-th component of the mean vector at normalized time $\tilde{t}$ is given in the quadratic case ($m = 3$) by $\mu_{ti} = b_{i1} + b_{i2}\tilde{t} + b_{i3}\tilde{t}^2$, where b_{ij} is an element of the coefficient matrix B. In general, $\mu_i = Bz_i$ where z_i is the i-th column of Z_L.

The two proposed approaches differ in the representation of time. Gish and Ng define the observed segment to be a linear sampling of a complete trajectory, so $\tilde{t} = (i-1)/(l-1) \in [0, 1]$ and the segment mean is comparable to a codeword in the Shiraki-Honda [16]. Deng *et al.* use absolute time, so $\tilde{t} \propto j$ for the jth frame in the segment, i.e. the trajectory varies with segment length. Using absolute time has the advantage of efficient recognition and segmentation algorithms, since the Markov assumption holds within and across segments, but it is only reasonable for sub-phonetic units (a phone of length l generally does not correspond to the first half of a phone of length $2l$). Since we hope to model units that are bigger than phones, we will follow the Gish-Ng approach [19].[1] The results of both sets of researchers for constant, linear and quadratic mean functions and context-independent phone modeling suggest that there is little to be gained by going beyond linear models, with the possible exception of modeling diphthongs. However, these experiments were based on phone and sub-phone acoustic modeling.

[1]The estimation equations given here will differ slightly from those in [19], since we use a transposed definition of Y_i and have not included the segmental mixture terms.

To estimate the parameters of this parametric segment model, which are B and Σ, assuming the Gish-Ng time warping [19], define the ML estimate of the trajectory parameter matrix for a single segment observation Y_i as

$$\beta_i = Y_i Z_{l_i}^t [Z_{l_i} Z_{l_i}^t]^{-1}, \tag{3}$$

and the segment level sample covariance as

$$\Sigma_i = \frac{1}{l_i} \sum_{j=1}^{l_i} (y_{i,j} - \beta_i Z_{l_i})(y_{i,j} - \beta_i Z_{l_i})^t. \tag{4}$$

The triple (β_i, Σ_i, l_i) together form a sufficient statistic for segment Y_i when the model regression order is m, and are a more efficient representation of the segment if $l_i > D + m$. Taking this statistic for all segments Y_i that map to the specific model of interest, the new model parameters are:

$$\hat{B} = \left[\sum_{i \in \mathcal{I}_a} \beta_i Z_{l_i} Z_{l_i}^t\right] \left[\sum_{i \in \mathcal{I}_a} Z_{l_i} Z_{l_i}^t\right]^{-1} \tag{5}$$

$$\hat{\Sigma} = \frac{1}{N} \sum_{i \in \mathcal{I}_a} l_i \Sigma_i + \frac{1}{N} \sum_{i \in \mathcal{I}_a} l_i (\beta_i - \hat{B}) Z_{l_i} Z_{l_i}^t (\beta_i - \hat{B})^t \tag{6}$$

where $\mathcal{I}_a = \{i : a_i = a\}$, $N = \sum_{i \in \mathcal{I}_a} l_i$.

2.3 Non-Uniform Unit Training

In this work, we extend the unit learning algorithm described in [11, 3] to handle segmental models, and add an iterative retraining step after clustering similar to that proposed in [16]. Unlike other work, we use the maximum likelihood objective function together with the segmental model distribution assumption in both clustering and iterative model re-estimation.

Initial Segmentation Algorithm

The initial segmentation algorithm is a slightly different implementation of the general approach described in [11]. Define $\delta_t(n)$ to be the log probability of the most likely segmentation for observations $y_1^t = \{y_1, \ldots, y_t\}$ and including n segments:

$$\delta_t(n) = \max_{l_1^n} \log \mathrm{p}(y_1^t, l_1^n). \tag{7}$$

The traceback information is stored in $\psi_t(n)$, which contains the ending time of the best previous segment that led to $\delta_t(n)$. Since the likelihood of a segment is determined by its own mean trajectory and not some fixed model set, shorter segments will always be more likely than longer segments and the maximum likelihood segmentation will be biased to use the minimum segment length as often as possible. In order to find a non-trivial segmentation, we impose a constraint that the resulting average (per frame) segmentation error must be greater than d_{min}. (We empirically test different values of d_{min} to determine a threshold to give segments with average length equal that of an average phone.) Then the dynamic programming segmentation algorithm involves:

Initial Segmentation Algorithm

```
Initialize: $t = l_{min}$
    $\delta_t(1) = \log \mathrm{p}(y_1, \ldots, y_t | l) \quad \forall i \in \mathcal{A}_1, \ l = l_{min}$
Iterate: $t = l_{min} + 1, \ldots, T; \forall n < t/l_{min}$
    $\delta_t(n) = \max_{t-l_{max} \le \tau \le t-l_{min}} \delta_{\tau-1}(n-1) + \log \mathrm{p}(y_\tau, \ldots, y_t | l_\tau); \quad l = t - \tau + 1$
    $\psi_t(n) = \operatorname{argmax}_{t-l_{max} \le \tau \le t-l_{min}} \delta_{\tau-1}(n-1) + \log \mathrm{p}(y_\tau, \ldots, y_t | l_\tau); \quad l = t - \tau + 1$
Traceback:
    $N^* = \operatorname{argmax}_n \delta_T(n) \quad \text{such that} \ \frac{1}{N^*}\delta_T(N^*) > d_{min}; \quad t = T, n = N$
    Iterate while $t > 0$:  1) $t' = \psi_t(n)$,  2) $t \leftarrow t', n \leftarrow n - 1$
```

The probability term $\log \mathrm{p}(y_\tau, \ldots, y_t | l_\tau)$ is based on the assumption of a parametric mean segment model using Equation 2 with the maximum likelihood parameters for the observed data $[y_{\tau+1}, \ldots, y_t]$. The particular estimate of the mean will depend on the order of the regression model. Since the segment lengths will almost always be too short to estimate an invertible covariance matrix, we will assume that the covariance is fixed for all segments in an utterance and use either the covariance estimated from all cepstra in the target utterance or from the full training set. It is important that some reasonable covariance matrix is used, as opposed to simply using Euclidean distance, since otherwise the dimensions of y_t with the highest variance (e.g. energy) will dominate the segmentation criterion.

Clustering

Once the training data has been segmented, the next step is to cluster the segments to obtain a fixed number of models. (The particular number of models will be chosen heuristically, to be comparable to an HMM context-dependent recognition system for experimental assessment.) Clustering involves initial codebook design, in which we use a binary splitting algorithm, followed by an iterative re-estimation algorithm, similar to the standard VQ design algorithm [15] except for the use of the maximum likelihood design criterion.

1. *Binary splitting initialization:* Obtain a set of K segment models by successively partitioning subsets of the training data.
   (a) Compute the mean and covariance for a single segment model to represent the entire training set. Set this model as "the model to be split".
   (b) Run a two-model clustering procedure (analogous to a two-codeword VQ design procedure) for the model to be split using only the segment observations assigned to that model, following the steps outlined in the full iterative re-estimation algorithm below. Increment the total number of models by 1.
   (c) If the desired number of models K has been reached, then stop. Otherwise, choose a new model to be split by finding the model with the largest average variance per frame, and go to step 1b.
2. *Iterative re-estimation:* Given an initial model set of the appropriate size K, then iterate:
   (a) Partition the training data, assigning each segment Y_i to $\mathcal{Y}_j$ according to

$$S(i) = \operatorname*{argmax}_{0<j\le K} \log \mathrm{p}(Y_i | \theta_j)$$

   so that $\mathcal{Y}_j$ is the set of segments that best fit model j.

(b) Re-estimate the model parameters using the maximum likelihood estimate for $j = 1, \ldots, K$

$$\theta'_j = \underset{\theta}{\operatorname{argmax}} \sum_{Y_i \in \mathcal{Y}_j} \log \mathrm{p}(Y_i|\theta)$$

according to Equations 3-6.

(c) Test to see if the relative increase in likelihood is below some threshold or if some maximum number of iterations has been reached. If so, then stop; otherwise go to step 2a.

The clustering algorithm described here is similar to that outlined in [1], with the exception that instead of the Itakura-Saito distortion measure we use maximum log likelihood under the assumption of a multivariate Gaussian with a time-varying mean specified by a polynomial function of a particular order.

Iterative Model Re-estimation

The iterative model re-estimation algorithm proposed here is similar to that described in [16], except that the objective function is maximum likelihood rather than minimum distortion. Iterative model re-estimation can also be seen as a Viterbi training procedure (i.e. the most-likely-state-sequence (MLSS) training algorithm described in [18]). The difference between this algorithm and Viterbi training with phonetic units is that re-segmentation is not constrained to the word sequence because there is as yet no lexical mapping.

Define $\theta^{(p)}$ to be the segment model distribution parameters at iteration p, and define $S^{(p)}$ and $A^{(p)}$ to be the segmentation and associated model label sequence for the training data at iteration p. Then, the re-estimation algorithm involves iterating:

1. *Segmentation:* Jointly find the most likely segment label sequence and its associated segmentation $(A^{(p)}, S^{(p)})$ for the training data, given model $\theta^{(p-1)}$.

2. *Estimation:* Find the maximum likelihood parameters $\theta^{(p)}$, given segmentation $(A^{(p)}, S^{(p)})$.

3. *Test for convergence:* Test to see if the relative increase in likelihood is below some threshold or if some maximum number of iterations has been reached. If so, then stop; otherwise go to step 1.

This algorithm is analogous to vector quantizer design: the segmentation step is equivalent to the partitioning step in VQ design, and the estimation step is analogous to the centroid computation.

The estimation step is identical to the estimation step described in the clustering algorithm above, and the test for convergence is also the same. The segmentation step, however, requires more computation than the clustering re-partitioning step, since we not only want to reassign segments to models but also to allow segment boundaries and the total number of segments to change. As a consequence, the segmentation step is essentially a segment recognition problem, which can be solved using dynamic programming (or Viterbi decoding). Since we eventually plan to have some sequence constraints via the pronunciation networks, and because even in phone recognition (vs. word recognition) performance improves significantly with a phone ngram model, we will use a segment label bigram in the resegmentation. (In the initial pass, the label sequence information is not easily available, in which case we use a unigram.)

Because segment recognition can be very expensive, we have introduced several pruning mechanisms to speed up the resegmentation process. First, we restrict the number of segments in the hypothesis ending at time t to be within some fraction (e.g. $\pm$ 10%) of the number of segments at that time in the previous iteration. Second, we use beam search pruning to reduce the number of prior segment candidates that are considered in extending the hypothesis. In addition, we use the standard segment modeling technique of imposing minimum and maximum duration constraints as a function of the particular segment label. Below we outline the segmentation search using these constraints.

Let $\mathcal{N}_t = [n_p(t)(1-\epsilon), n_p(t)(1+\epsilon)]$ define the possible number of segments in a sequence ending at time t. Define $\mathcal{A}_t$ to be the set of active segment models candidates ending at time t, which is a pruned subset of the entire set $\mathcal{A}$ based on likelihood relative to the best segmentation ending at time t. Define $\rho(t, j)$ to be the set of allowable segment start times for a segment with label j ending at time t, which is determined by the minimum and maximum constraints on segment duration. Finally, define $\delta_t(n, i)$ to be the log probability of the most likely segmentation/label sequence ending with segment label i for observations $y_1^t = \{y_1, \ldots, y_t\}$ and including n segments:

$$\delta_t(n,i) = nK + \max_{l_1^n, a_1^{n-1}} \log \mathrm{p}(y_1^t, l_1^n, a_1^{n-1}, a_n = i),$$

where K is an insertion penalty added to provide an additional control on the number of resulting segmentations. The traceback information is stored in $\psi_t(n,i)$, which contains the ending time and label of the best previous segment that led to $\delta_t(n,i)$. Then the dynamic programming segmentation algorithm involves:

Re-Segmentation Algorithm

Initialize: $t = l_{min}$

$$\delta_t(1,i) = \log \mathrm{p}(y_1|l,i)\mathrm{p}(l|i)\mathrm{p}(i) \quad \forall i \in \mathcal{A}_1,\ l = l_{min}$$

Iterate: $t = l_{min}+1, \ldots, T;\ \forall i \in \mathcal{A}_t;\ \forall n \in \mathcal{N}_t$

$$\delta_t(n,i) = K + \max_{j \in \mathcal{A}_\tau, \tau \in \rho(t,i)} \delta_{\tau-1}(n-1,j) + \log[\mathrm{p}(y_\tau, \ldots, y_t|l_\tau, i)\mathrm{p}(l_\tau|i)\mathrm{p}(i|j)] \quad l_\tau = t - \tau + 1$$

$$\psi_t(n,i) = \operatorname*{argmax}_{j \in \mathcal{A}_\tau, \tau \in \rho(t,i)} \delta_{\tau-1}(n-1,j) + \log[\mathrm{p}(y_\tau, \ldots, y_t|l_\tau, i)\mathrm{p}(l_\tau|i)\mathrm{p}(i|j)] \quad l_\tau = t - \tau + 1$$

Traceback:

$$(N^*, \hat{a}_{N^*}) = \operatorname*{argmax}_{n \in \mathcal{N}_T, i} \delta_T(n,i), \quad t = T, n = N^*$$

Iterate while $t > 0$: 1) $(\hat{a}_{n-1}, t') = \psi_t(n, \hat{a}_t)$, 2) $t \leftarrow t', n \leftarrow n-1$

3 Mapping to Lexical Entries

Once the inventory of acoustic models is defined, the second main problem is to find a mapping from a sequence of these units to sequences of words. This mapping can be simply a linear string of sub-word units, or a network. To simplify the discussion, we will refer to these mappings as "pronunciations", even though they may not be based on phonetic units. In addition, the units may span word boundaries, so we will use the terms "lexical item" and "lexicon" loosely, to include single orthographic words or sequences of words. Since the lexicon is not a simple list of words, the first step will be to determine the lexicon, and then the mapping from segment model labels to lexical entries can be derived. The proposed algorithms for these two problems are summarized in this section.

3.1 Determining the Lexicon

A problem faced in our work, not addressed by any previous work, is that the automatically learned units may span word boundaries, so some pronunciations will be defined for word sequences as opposed to single words. One could constrain the unit learning process to enforce segment boundaries at word boundaries, but such a constraint might limit the potential of the model. For example, if "did you" is frequently said as "/d/ih/jh/ax/" it might be best to represent as a single lexical item. Another solution is to allow multi-word entries in the lexicon, but when all possible multi-word entries are included in the lexicon, the lexicon size becomes so large that the search will become too costly. Therefore, the the problem becomes one of finding a minimal (limited) set of words and phrases that best represent the cross-word phonetic assimilation in the data.

The approach we propose for building the lexicon is to start with a lexicon of all possible word sequences with segments that span word boundaries, and progressively prune infrequent word sequences out while adding segment boundary constraints. The basic algorithm is illustrated in Figure 1, and is summarized in more detail below.

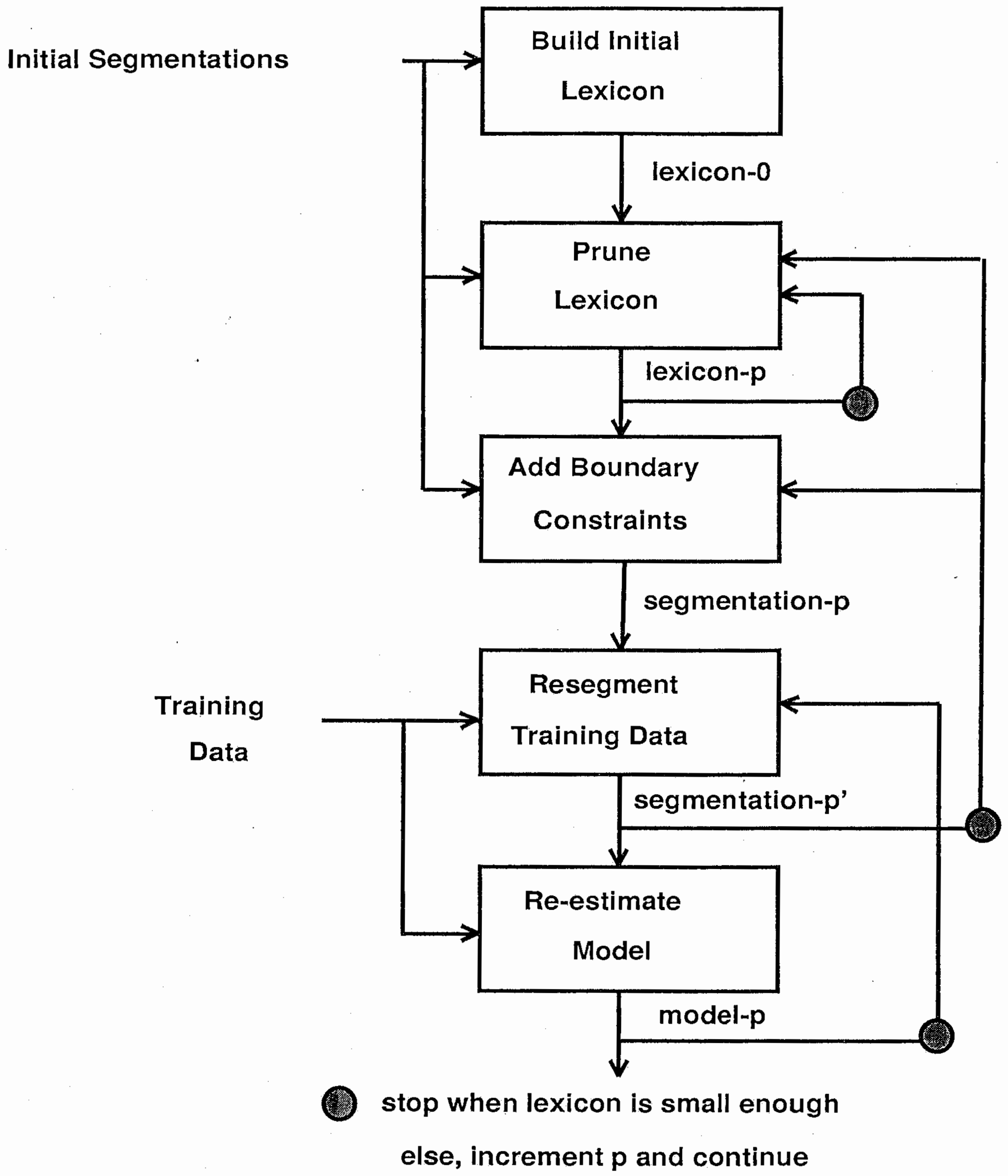

図 1: Block diagram illustrating the lexicon design process.

Given an inventory of models $\theta^{(0)}$, training data marked with canonical word boundaries (labeled by hand or by Viterbi alignment using a phonetic word pronunciation) and segment boundaries for the non-uniform units, a single word lexicon, and a maximum lexicon size, then iterate:

1. *Initialize:*
   (a) If an automatic segmentation boundary is within k frames of the canonical word boundary (e.g. $k = 1$), move and fix the boundary at the canonical boundary location.
   (b) Build an initial lexicon, consisting of all single-word entries and multi-word entries found in the training data with segments spanning all word boundaries within the sequence. A segment "spans" a word boundary if the time difference from the canonical word boundary to the closest segment boundary is more than T ms.
2. *Prune lexicon:*
   Compute a score for each multi-word sequence in the lexicon that is a measure of cross-word phonetic assimilation, and is also based on the relative frequency of the word sequence and cross-word segments. Then, prune all multi-word sequences from the lexicon with a low score (stopping if the desired lexicon size is reached).
3. *Add boundary constraints:*
   For all instances of pruned word sequences that have segments that span the word boundary, introduce a new boundary at the canonical word boundary time.
4. *Segmentation:*
   (a) Resegment the training data using the current model inventory, retaining any boundary constraints from a previous iteration.
   (b) Move and fix all boundaries that are close to canonical word boundaries, as in step 1a. For each boundary that has been moved, if the word sequence at that boundary appears nowhere else with a segment spanning the boundary, then remove this word sequence from the lexicon.
5. *Model re-estimation:*
   Re-estimate the parameters of the models $\theta^{(p)}$ according to Equations 3-6.
6. *Stopping test:*
   If the lexicon is no bigger than the maximum size, then stop. Otherwise, increment p and go to step 2.

Critical to the success of this algorithm is the mechanism for eliminating word pairs in Step 2 above, and there are a variety of options. For example, one could simply eliminate the word sequence with the lowest frequency of actual multi-word pronunciations. Alternatively, one could also consider the frequency and/or length of the segment that spanned that boundary. In the experiments reported here, the score in step 2 is computed as the time difference between the canonical word boundary time c and the closest automatic boundary time $A(c)$ summed over all word boundaries in the multi-word sequence:

$$S(e) = \frac{1}{I(e)} \sum_{i \in \{e\}} \sum_{c \in B(i)} |c - A(c)| \tag{8}$$

where $S(e)$ denotes the score for lexical entry e, $I(e)$ denotes the number of word boundaries within lexical entry e (i.e. a 3-word sequence would have $I(e) = 2$) , $\{e\}$ denotes the instances of lexical entry e in the training data, and $B(i)$ denotes the phone-based word-boundary times of instance i. Note that the score has a bias toward frequent multi-word sequences.

By applying this algorithm, we obtain a lexicon consisting of all distinct single word entries found in the training data plus the most frequently reduced word sequences which are better modeled by a multi-word lexical entry. A possible problem with this approach is that the additional segmentation passes may be costly. However, by imposing word boundary constraints on the segmentation, which are needed for the algorithm anyway, computation is significantly reduced relative to the resegmentation step described in Section 2.3. In addition, we can reduce the size of the lexicon in stages, repeating steps (2)-(3) to minimize the changes in the acoustic units at each stage.

3.2 Pronunciation Modeling: Previous Work

Once we are given a lexicon word list, then the remaining task is to find the pronunciation network for each lexical entry. A variety of approaches have been proposed for deriving word "pronunciations", which can be related in terms of algorithmic structure independently of whether they are phonetically based or based on automatically learned units. Rather than doing a historical review, in this section we will outline three general approaches that have been taken and explain how they are or are not appropriate to the problem of interest here. In addition, we discuss possible probabilistic pronunciation modeling assumptions within the context of these approaches and associated trade-offs for the problem of mapping from unconstrained acoustic sub-word units to words.

To describe a general approach to lexical mapping design, we begin by introducing some notation. Let $Y = (y_1, \ldots, y_T)$ correspond to a sequence of acoustic observations for a word, where y_t is a member of this sequence and is a d-dimensional vector, and let $\mathcal{Y}_W$ represent the set of all such sequences in the training data corresponding to word W. Let $\phi = (\phi_1, \ldots, \phi_n)$ represent a sequence of labels of acoustic units (e.g. phones or automatically learned sub-word units) which we call a "pronunciation" of a word. The symbolic labels $\phi_i \in \Phi$ form an intermediate representation between words and acoustics that we can use to make more reasonable conditional independence assumptions, i.e. a subsequence of observations is conditionally independent of other observations given the model label ϕ_i. $W = (g_1, \ldots, g_m)$ will be used to represent a word, where $g_i \in \mathcal{G}$ might be either a letter or an element of a phonetic baseform from a single-pronunciation dictionary. Note that each of these representations of a word – acoustic (y_1^T), symbolic acoustic (ϕ_1^n), and dictionary (g_1^m) – have different lengths. In order to estimate a mapping, we could potentially use two model components:

- $p(Y|\phi)$: an *acoustic model,* such as an HMM or the segment model described in Section 2.2, and
- $p(\phi|W)$: a *pronunciation model,* which may be a letter-to-phone probabilistic mapping if W is represented by letters and ϕ_i is a phone, but may also be based on non-phonetic units ϕ_i.

Using these definitions, there are essentially three types of approaches that have been proposed in the literature, as described below.

The first approach aimed at finding a single pronunciation, i.e. a linear sequence of sub-word units, according to

$$\begin{aligned}
\hat{\phi}(W) &= \operatorname*{argmax}_{\phi} \log \mathrm{p}(\phi|\mathcal{Y}_W, W) && (9)\\
&= \operatorname*{argmax}_{\phi} \log [\mathrm{p}(\mathcal{Y}_W|\phi)\mathrm{p}(\phi|W)] && (10)\\
&= \operatorname*{argmax}_{\phi} \left[\log \mathrm{p}(\phi|W) + \sum_{Y_i \in \mathcal{Y}_W} \log \mathrm{p}(Y_i|\phi)\right] && (11)
\end{aligned}$$

In equation 10, we make the assumption that the acoustic observations are conditionally independent of the word given the specific pronunciation, which is a standard assumption made in speech recognition work. This approach was proposed as described above in [12], but it can also be thought of as more sophisticated versions of earlier work [2, 3] that used simple models for $\mathrm{p}(\mathcal{Y}|\phi)$ and $\mathrm{p}(\phi|W)$. For example, in the deterministic mapping (method 3) described in [3], the distribution $\mathrm{p}(Y_i|\phi)$ corresponds to a time-warped distance and all possible pronunciations ϕ for word W are assumed to be equally likely. In [12], the acoustic model is an HMM, and the pronunciation model uses a decision tree estimator of the probability of a phone given a sequence of letters. In [21], the acoustic model is an HMM, but the pronunciation model is given by a baseform predicted by a synthesizer (Dectalk) expanded using a confusion matrix of synthesizer prediction errors. Note that this approach finds a pronunciation for each word based on acoustic observations of that word. If a word in the lexicon is not observed in the training data, then the pronunciation must be derived from the pronunciation model only:

$$\hat{\phi} = \operatorname*{argmax}_{\phi} \log p(\phi|W), \tag{12}$$

which was first proposed using an HMM model of pronunciation by Lucassen and Mercer [22]. When acoustic observations are available, it is beneficial to use both sources of information [2].

A disadvantage of the above approach is that the pronunciation model, if trained from a dictionary, may not reflect the types of pronunciation variation present in the particular type of speech to be recognized. For example, it may not capture all of the different types of reduction phenomena in spontaneous speech. This is not a problem if there are a large number of acoustic training tokens per word, but could be a problem for infrequently observed words. Thus, a second approach is to estimate the pronunciation model based on the training data. Specifically, assume an acoustic model $p(Y|\phi)$ (an HMM in previously reported work) and an initial pronunciation model $p(\phi|W)$ and then execute the following steps:

1. Segment the training data for each instance of a word, Y^i, to find the observed pronunciation for that instance of that word
$$\hat{\phi}^i = \operatorname*{argmax}_{\phi} p(Y^i|\phi)p(\phi|W).$$
2. Use the pairs $\{(\hat{\phi}^i, W^i)\}$ from the entire training data to estimate a new pronunciation model $\tilde{p}(\phi|W)$.
3. Use $\tilde{p}(\phi|W)$ to find a new pronunciation network for each word W. For example, if a single pronunciation is desired, then
$$\hat{\phi}(W) = \operatorname*{argmax}_{\phi} \tilde{p}(\phi|W).$$
Alternatively, use $\tilde{p}(\phi|W)$ to expand a baseform to include all non-zero probability pronunciations.

This algorithm, which can optionally be iterated, was initially proposed by Cohen [23], who used hand-written phonological rules to derive an initial pronunciation network from a phonetic baseform, and then trained the probabilities of those rules, effectively giving $\tilde{p}(\phi|W)$. The same general approach was used by Ljolje *et al.* [24], where the pronunciation model used decision trees to model the phone probabilities given a baseform representation of the word [25]. Their initial pronunciation model was trained on the hand-labeled TIMIT corpus. Paliwal's stochastic lexical building algorithm [3] is another example, where $\tilde{p}(\phi|W)$ is essentially a bigram model of the sequence of subword units.

Finally, the third approach uses no pronunciation model at all, but rather uses structural inference techniques to either expand a baseform or compress a large pronunciation network to maximize coverage of the observed pronunciations and minimize the size of the network, as in [1, 26, 27]. A related technique, described in [28], looks at recognition errors rather than forced alignments in expanding the network, though this approach requires that some initial baseform be available. All of these structural inference techniques suffer from the problem that they cannot easily be used for unobserved words and may not be robust for infrequently observed words; hence we will not consider them as an option for this work.

Although the structural inference approach is inappropriate given our need to find pronunciations for unobserved words, both other approaches have advantages that we can benefit from. The first method provides the best solution for words which are well trained, since the pronunciation is based directly on the observed data for specific words. The second method, on the other hand, uses a more appropriate $\tilde{p}(\phi|W)$ that is tuned to the recognition task, and so will provide a better solution for infrequently observed words. As a consequence, we propose a hybrid of these two approaches in Section 3.3 for our work.

A superficial difference between the first two methods is whether a single pronunciation is derived, as in Equation 11, versus a pronunciation network. In fact, both methods can be used to either generate single pronunciation strings, as shown by Equation 12, or multiple pronunciation by using an "N-best" style maximization search [29] in place of the simple maximization in Equation 11 to obtain a pronunciation lattice. The more important question is to what extent multiple pronunciations make sense. A problem with using large pronunciation networks is that they increase the acoustic confusability of words, since it is more likely that two different words will have a common pronunciation. The phone transition probabilities compensate somewhat for this problem by providing pronunciation probabilities, but they are a small part of the overall word score relative to the observation likelihoods. For this reason, there are numerous (unreported) anecdotal results showing no benefit to using multiple pronunciations, and most recognition systems use very few pronunciations per word. Others, however, find that good results can be obtained by appropriate estimation techniques and pronunciation network pruning [23, 27, 24]. Pruning techniques vary from simple elimination of low probability arcs in the network [30] to elimination of low probability phonological rules. Certainly in modeling spontaneous speech, it is likely to be beneficial to allow for some pronunciation variability.

An important issue associated with the above approaches is the form of the pronunciation model, or $p(\phi|W)$. As already mentioned, decision trees are used in [4, 24]. Decision trees act essentially as variable-order n-gram models, and thus are more powerful than bigrams. An alternative model is the hierarchical structure proposed by Meng *et al.* [31], which puts probabilities on the sister transitions within a level of a "parse tree" as defined by a probabilistic grammar. An advantage of the grammatical model is that it provides a mechanism for encoding linguistic knowledge about morphological, syllabic and phonetic structure, among other levels. A disadvantage is that the grammar rules must be written by hand, although it is automatically trained and used as a probabilistic mapping, and it may be difficult to write grammatical rules to describe units that are not tied in anyway to phones or syllables.

3.3 Pronunciation Network Estimation

¿From lessons learned in previous research, we believe that a pronunciation network design algorithm should have the following features:

- the resulting lexicon includes multiple pronunciations, but not too many per word;
- the design algorithm uses all acoustic observations where possible; and
- the design algorithm uses a task-dependent pronunciation model for generalizing to infrequently observed or unobserved cases.

To meet these objectives, we will use a hybrid version of the methods described in Section 3.2, which uses training data to determine word-specific pronunciations where there is sufficient data to do so and a robust general pronunciation model otherwise.

In order to make the difference between this approach and previous work clear, it will be useful to review the notation from Section 3.2. As before, $Y = (y_1, \ldots, y_T)$ corresponds to a sequence of acoustic observations for a word, but now we will let the intermediate representation be the labels ϕ_i take on values $a \in \mathcal{A}$, which are the indices of our automatically learned segmental units. In addition, we will use an expanded notion of a word, $W = (g_1, \ldots, g_m, F_1(W), \ldots, F_k(W))$, where $F_i(W)$ are features of the word, such as the part-of-speech tag. To simplify the pronunciation model, we will assume that g_i are phonetic symbols available from a single pronunciation dictionary, rather than the letters in the orthographic transcription of the word, since learning pronunciations directly from the letters will likely require a more complex mapping and therefore more training data than learning from a phonetic baseform. As before, the three different representations of a word have different lengths.

The hybrid algorithm, outlined below, first estimates a general probabilistic pronunciation model $p(\phi|W)$ from the training data, and then uses this model to find a smaller pronunciation network $\hat{\phi}(W)$ for each individual word. These two steps can optionally be iterated, in combination with re-segmentation of the training data, to reduce the size of the pronunciation network gradually.

Lexical Mapping Derivation

1. **Pronunciation model estimation:** Use the word boundaries and segmentations to estimate a pronunciation model $\mathrm{p}(\phi|W)$, to cover all words in the lexicon.

2. **Pronunciation network generation:** Using the notation LattArgmax to indicate a maximum likelihood search which returns a lattice of hypotheses

   - If the number of training tokens for W is greater than N

$$\hat{\phi}(W) = \underset{\phi}{\mathrm{LattArgmax}} \left[\log \mathrm{p}(\phi|W) + \sum_{Y^i \in \mathcal{Y}_W} \log \mathrm{p}(Y^i|\phi) \right]$$

   - Else

$$\hat{\phi}(W) = \underset{\phi}{\mathrm{LattArgmax}} \left[\log \mathrm{p}(\phi|W)\right].$$

3. **Optional iteration:** If desired, resegment the training data using $\{\hat{\phi}(W)\}$ and go to step 1.

The maximization to obtain a lattice in step 2 above requires some criterion for determining the size of the lattice. The criterion can either be a constraint on the number of N-best pronunciations, which are then compressed into lattice form, or it can be a constraint on the likelihood of a pronunciation string relative to the most likely pronunciation for that word. Given this criterion, a two-pass (forward and backward) search algorithm is used to obtain a lattice for each word, which takes into account the overall path probability and thus gives a better result than simply pruning low probability arcs or rules.

In initial work, we simplify the problem by omitting the word features $F(W)$, and use a general stochastic model for $\mathrm{p}(\phi|W)$ that is similar to the work described in [22, 25] in that the acoustic unit labels are treated as observations from an underlying state sequence. The state sequence, which in our case is the baseform g_1^m, may or may not be hidden depending on whether there are multiple phonetic pronunciations for a word. Our work differs from previous work primarily in the use of automatically learned acoustic units rather than phonetic units, with the associated problem of greater variability in the observations associated with a word. Let g_1^m denote the baseform pronunciation of a word, which is given by a phonetic dictionary, and $\phi = \phi_1^n$ an automatic unit label sequence. Ignoring the word features for the moment, we use $\mathrm{p}(\phi|W) = \mathrm{p}(\phi|g_1^m)$ as the probability of a particular automatic label "pronunciation" ϕ_1^n and baseform pronunciation g_1^m. Let α_i be the (possibly empty) subsequence of ϕ_t that are associated with g_i. Then the pronunciation probability is

$$\begin{aligned} \mathrm{p}(\phi_1^n, g_1^m) &= \sum_{\{\alpha_1^m\}} P(\phi_1^n, g_1^m, \alpha_1^m) \\ &= \sum_{\{\alpha_1^m\}} P(\alpha_1^m|g_1^m) P(g_1^m) \end{aligned} \tag{13}$$

where α_1^m contains all the information in ϕ_1^n. The set of all possible partitionings of ϕ_1^n into subsequences is denoted by $\{\alpha_1^m\}$. Assuming that the baseform sequence g_1^m is Markov and that the α_i's are conditionally independent given the "states" g_i, the probability of a particular observed pronunciation of word ϕ_1^n is

$$\mathrm{p}(\phi_1^n, g_1^m) = \sum_{\{\alpha_1^m\}} \prod_{i=1}^{m} \mathrm{p}(\alpha_i|g_i)\mathrm{p}(g_i|g_{i-1}). \tag{14}$$

If there is a single baseform that is known for all words, then the sequence g_1^m is known and the Markov probabilities are dropped. In [25], $\mathrm{p}(\alpha_i|g_i)$ is represented separately for all possible subsequences α_i, since the total number is small (the observed subsequence lengths $l(\alpha_i)$ are limited to be ≤ 2 with additional restrictions on the number of length 2 sequences). In our case, on the other hand, $l(\alpha_i)$ can vary substantially, so some simplifying assumptions are needed to reduce the parameter space. One solution, implemented in our initial work, is to use phone-dependent unit label n-grams, e.g.

$$\mathrm{p}(\alpha_i|g_i) = \mathrm{p}(l(\alpha_i)|g_i)\mathrm{p}(\phi_{t(\alpha_i)}|g_i) \prod_{t=t(\alpha)+1}^{t(\alpha)+l(\alpha)-1} \mathrm{p}(\phi_t|\phi_{t-1}, g_i) \tag{15}$$

where $t(\alpha_i)$ is the time of the first acoustic unit label in α_i and we use the fact that $\mathrm{p}(\alpha_i|g_i) = \mathrm{p}(\alpha_i, l(\alpha_i)|g_i)$. The extension of this model to handle word features is facilitated by the use of decision trees, which reduces the number of parameters in the model. In equation 15, we simply replace $\mathrm{p}(\phi_t|\phi_{t-1}, g_i)$ with $\mathrm{p}(\phi_t|T(\phi_{t-1}, g_i, F(W)))$, where $T(\cdot)$ is a decision tree mapping.

This model is essentially a discrete distribution segment model, so the parameters can be trained using the generalized EM or the MLSS algorithms described in [18]. Since the model is a generalization of an HMM, HMM lattice search algorithms can be used for finding the pronunciation network with some modifications to handle variable-length observations.

4 Word Recognition Search Algorithm

The goal of any word recognition algorithm based on a statistical model is to find the most likely word sequence $\hat{w}_1^K = \{w_1, \ldots, w_K\}$ given an observation sequence for an entire utterance $y_1^T = \{y_1, \ldots, y_T\}$, or

$$\hat{w}_1^K = \operatorname*{argmax}_{w_1^K, K} \mathrm{p}(w_1^K|y_1^T), \tag{16}$$

where $w_k \in \mathcal{W}$ and $|\mathcal{W}|$ is the number of words in the vocabulary. Using the models described in the previous section, the problem becomes

$$\hat{w}_1^K = \operatorname*{argmax}_{w_1^K, K, \phi_1^M, M} \mathrm{p}(y_1^T|\phi_1^M)\mathrm{p}(\phi_1^M|w_1^K)\mathrm{p}(w_1^K) \tag{17}$$

where ϕ_1^M is a sequence of acoustic units, $\mathrm{p}(y_1^T|\phi_1^M)$ is given by the acoustic segment models, $\mathrm{p}(\phi_1^K|w_1^T)$ is given by the pronunciation model, and $\mathrm{p}(w_1^K)$ is given by the language model. The differences relative to the HMM recognition problem, which has analogous component models, is that the acoustic model and the pronunciation model are more complex and effectively operate in a higher dimensional state space. (The language model is not addressed as a research problem here; we shall simply use the standard n-gram language modeling techniques.) As a result, the cost of the recognition search is much higher, and some computation reduction compromises must be made. Below, we describe two approaches for reducing computation, both of which require a multi-pass decoding strategy. First, we can reduce the number of candidate acoustic unit labels in a first pass of recognition that ignores the language and pronunciation models, and then impose these constraints in a second pass. Alternatively, we can use an HMM search to narrow the candidate word hypotheses to an N-best list or a word graph, and then proceed with recognition directly from the cepstra. Both algorithms are similar in structure, so we begin with some notational preliminaries defining the decoding network structure.

4.1 Network Structure

The possible words in the vocabulary are $w \in \mathcal{W}$, $|\mathcal{W}| = V$, and each word has a pronunciation baseform $\{g_1^{(w)}, \ldots, g_{|w|}^{(w)}\}$ where $g_i^{(w)} \in \mathcal{G}$ is a phone and $|w|$ denotes the length in phones of word w. To simplify the discussion, we will assume that the grammar allows all possible word sequences and that their probability is represented by a bigram language model:

$$\mathrm{p}(w_1, \ldots, w_k) = p(w_1| < beg >)[\prod_{i=2}^{k} p(w_i|w_{i-1})]p(< end > |w_k) \tag{18}$$

where $< beg >$ and $< end >$ are symbols for the beginning and end of the utterance.

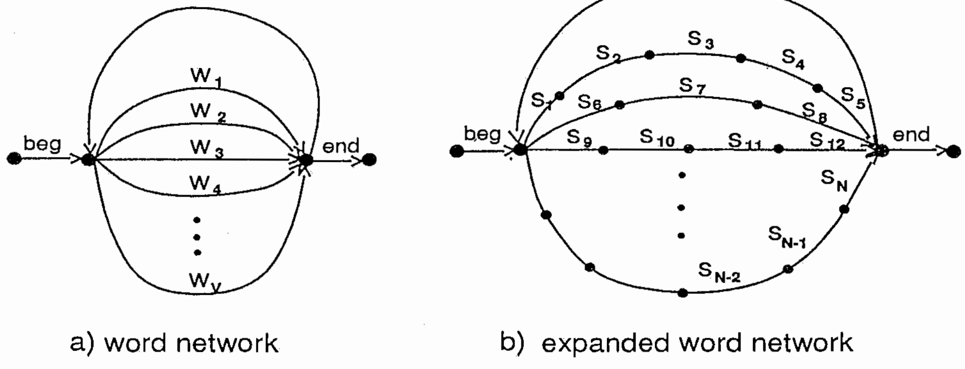

図 2: Word network for recognition search and associated expanded phone network.

The pronunciation baseform can be thought of as "hidden states", with the observable acoustic segments $\{\phi_i\}$ being the output of these states and the cepstral features $\{y_t\}$ being the output observations associated with the acoustic segments. From a generative perspective, we have:

$$\{w_k\} \rightarrow \{g_i\} \rightarrow \{\phi_j\} \rightarrow \{y_t\} \tag{19}$$

(Note that, in this view, automatically learned units $\phi_j \in \mathcal{A}$ can only span multiple phones if some phones have null outputs.) The likelihood of an M-length acoustic unit sequence ϕ_1^M and $|w|$-length phone sequence $g_1^{|w|}$ given a deterministic mapping of the acoustic unit sequence to the phone sequence is given by:

$$\mathrm{p}(\phi_1, \ldots, \phi_M, g_1, \ldots, g_{|w|}|w) = \prod_{l=1}^{|w|} p(\alpha_l|g_l)p(g_l|g_{l-1}, w) \tag{20}$$

where $\alpha_1^{|w|}$ denotes the mapping of the acoustic segment sequence ϕ_1^M to the phone sequence $g_1^{|w|}$ and α_l denotes the (possibly empty) acoustic segment subsequence that maps to phone g_l.

For recognition decoding, all the words are put in parallel with a loop from the endings of words, as illustrated in Figure 2(a). Then that network is expanded in terms of the phonetic baseforms of the words. Figure 2(b) illustrates the expansion for the case where w_1 has a 5-phone baseform, w_2 has a 3-phone baseform, etc. Each arc in the expanded network is separately labeled s_p, where $s_p \in \mathcal{G}$ and there are N arcs in the full network. Typically, for large vocabularies, $N >> |\mathcal{G}|$. The set $\{s_p\}$ represents all possible phones in all possible words. If the same phone appears in multiple words, then it will correspond to multiple s_p instances. Note that only a subset of arcs s_p are allowed to start the utterance, and another subset is allowed to end utterances.

4.2 Recognition from Acoustic Unit Labels

One (sub-optimal) approach to recognition search is to first find the most likely acoustic segment sequence $\{\phi_1, \ldots, \phi_M\}$, and then use this sequence as the input to the recognition search. The most likely acoustic segment sequence can be found using the re-segmentation algorithm described in Section 2. Assuming the sequence ϕ_1^M is given, the recognition problem becomes

$$\hat{w}_1^K = \operatorname*{argmax}_{w_1^K, K} \mathrm{p}(w_1^K)\mathrm{p}(\phi_1^M|w_1^K) \tag{21}$$

$$= \operatorname*{argmax}_{w_1^K, K} \prod_{k=1}^{K} \mathrm{p}(\phi(w_k)|g(w_k))\mathrm{p}(w_k|w_{k-1}), \tag{22}$$

where $\phi(w_k)$ is the subsequence of ϕ_j that map to w_k and $g(w) = \{g_1^{(w)}, \ldots, g_{|w|}^{(w)}\}$ is the phonetic spelling of w_k. The solution requires dynamic programming to find the mapping between the acoustic units and the word pronunciation, which builds on the notion of g_i as a hidden state.

Let $\delta_1(n,m,q)$ denote the log probability of the most likely phone sequence that is allowable by the grammar and ends with the phone corresponding to arc s_n, using the best possible alignment to the sequence $\phi_1,\ldots,\phi_m$. When $q=0$, there is a null output associated with the last phone s_n, and there is at least one acoustic segment for $q=1$:

$$\delta_1(n,m,0) = \max_{g_1^{i-1},i} \log p(g_1,\ldots,g_{i-1},g_i=s_n,\phi_1,\ldots,\phi_m,\alpha_i=\emptyset) \tag{23}$$

$$\delta_1(n,m,1) = \max_{g_1^{i-1},i} \log p(g_1,\ldots g_{i-1},g_i=s_n,\phi_1,\ldots,\phi_m,\alpha_i\neq\emptyset). \tag{24}$$

We use s_n rather than some $g\in\mathcal{G}$ because we need to constrain the phone sequence by the word grammar. We will recursively update $\delta_1(n,m,0)$ and $\delta_1(n,m,1)$ and then trace back the phone sequence at the end of decoding. This phone sequence will correspond to a unique word sequence because of the grammatical constraint. Define $\Pi(s_n)$ as the set of arcs in the network that are allowed to precede s_n, and $\mathcal{N}(m)=\{s_n : g_m=s_n\}$ as the allowable arcs for segment label g_m.[2] Let K_{max} be the maximum number of acoustic units that can map to a phone. The search then involves:

```
For $m = 1,\ldots,M$
    For $n \in \mathcal{N}(m)$
```

$$\begin{aligned}
\delta_1(n,m,0) &= \max_{s_{n'}\in\Pi(s_n),q\in\{0,1\}} [\log \mathrm{p}(\emptyset|s_n)+\delta_1(n',m,q)+c(n',n)]\\
\psi_1(n,m,0) &= \operatorname*{argmax}_{s_{n'},q} [\log \mathrm{p}(\emptyset|s_n)+\delta_1(n',m,q)+c(n',n)]\\
\delta_1(n,m,1) &= \max_{K_{max}\le k<0,s_{n'}\in\Pi(s_n),q\in\{0,1\}} [\log \mathrm{p}(\phi_{m-k+1},\ldots,\phi_m|s_n)+\delta_1(n',m-k,q)+c(n',n)]\\
\psi_1(n,m,1) &= \operatorname*{argmax}_{k,s_{n'},q} [\log \mathrm{p}(\phi_{m-k+1},\ldots,\phi_m|s_n)+\delta_1(n',m-k,q)+c(n',n)]
\end{aligned}$$

The term $c(n',n)$ is the cost of going from $s_{n'}$ to s_n, which could be simply $\mathrm{p}(s_n|s_{n'})$ if that includes the word bigram probability, but it might also include word and/or phone insertion penalties. The term $\mathrm{p}(\phi_{m-l},\ldots,\phi_m|s_n)$ is the pronunciation model, which can be a simple n-gram, as we used in initial development, or a more sophisticated (e.g. decision tree) model. The function $\psi_1(n,m,q)$ allows you to do the traceback at the end of decoding, after you find the best final arc:

$$\operatorname*{argmax}_{n\in\mathcal{N}(M),q\in\{0,1\}} \delta_1(n,M,q).$$

If we assume the "observations" ϕ_j associated with a state s_n are conditionally Markov given s_n and incorporate self-loops s_n, then the search further simplifies and the non-null observation likelihood update formulas are

$$\delta_1(n,m,1) = \max_{s_{n'}\in\Pi(s_n),q\in\{0,1\};s_{n'}=s_n,q=1} [\log \mathrm{p}(\phi_m|s_n,\phi_{m-1})+\delta_1(n',m-1,q)+c(n',n)] \tag{25}$$

$$\psi_1(n,m,1) = \operatorname*{argmax}_{s_{n'},q} [\log \mathrm{p}(\phi_m|s_n,\phi_{m-1})+\delta_1(n',m-1,q)+c(n',n)]\,. \tag{26}$$

In this case, the length k of the substring of symbols ϕ_j is captured implicitly by the number of times s_n is visited, and there is not maximum imposed on this length.

4.3 Recognizing from Cepstral Observations

The problem of recognizing words directly from acoustic observations takes on the same form as in the previous case, except that all three terms in equation 17 must be computed and the algorithm must also be updated with each observation time step. Since the cost of the acoustic segment likelihood calculation can be

[2]Another possible definition of $\mathcal{N}(m)$ would be $\mathcal{N}(m)=\{s_n : g_m=s_n, g_{m-1}\in\Pi(s_n)\}$, which has the advantage that it is a smaller set but the disadvantage that it is a bit more work to find.

quite high, we will simplify the pronunciation model by making the Markov assumption, as in equations 25 and 26 above.

Let $\delta_2(t,n,a)$ denote the log probability of the most likely phone sequence that is allowable by the grammar and ends with the phone corresponding to arc s_n, using the best possible alignment to the observations $y_1, \ldots y_t$ with the best possible acoustic label sequence. In this case, we let $\emptyset$ be a possible value of a, which occurs when there is a null output associated with the last phone s_n.

$$\delta_2(t,n,a) = \max_{g_1^{i-1}, i, \phi_1^{m-1}, m} \log p(g_1, \ldots, g_{i-1}, g_i = s_n, \phi_1, \ldots, \phi_{m-1}, a, y_1, \ldots, y_t) \tag{27}$$

As before, we will recursively update $\delta_2(t,n,a)$ and then trace back the phone and associated word sequence at the end of decoding. In decoding from the observations, the constraints on n originate from the observations rather than the (unknown) acoustic units, so we define $\mathcal{N}(t)$ as the set of states allowable at time t (which would be controlled, for example, by a likelihood pruning threshold). In addition, we will define $\mathcal{A}(n)$ as the set of acoustic labels allowable as outputs from state s_n. As before, $\Pi(s_n)$ is the set of arcs in the network that precede s_n. Let L_{max} and L_{min} be the maximum and minimum number of cepstral frames in an acoustic unit. The search then involves:

```
For $t = 1, \ldots, T$
  For $n \in \mathcal{N}(t)$
    For $a \in \mathcal{A}(n)$
```

$$\begin{aligned}
\delta_2(t,n,\emptyset) &= \max_{s_{n'} \in \Pi(s_n), a' \in \mathcal{A}(n')} [\log \mathrm{p}(\emptyset | s_n, a') + \delta_2(t, n', a') + c(n', n)] \\
\delta_2(t,n,a) &= \max_{s_{n'} \in \Pi(s_n) \cup s_n, a' \in \mathcal{A}(n'), L_{max} \le l \le L_{min}} [\log \mathrm{p}(y_{t-l+1}, \ldots, y_t | a) + \\
&\qquad \log \mathrm{p}(a | s_n, a') + \delta_2(t-l, n', a') + c(n', n)] \quad \text{for } a \neq \emptyset \\
\psi_2(t,n,a) &= \operatorname*{argmax}_{s_{n'}, a', l}(\textit{above quantity})
\end{aligned}$$

As before, the function $\psi_2(t,n,a)$ allows you to do the traceback at the end of decoding, after you find the best final arc and acoustic unit:

$$\operatorname*{argmax}_{n \in \mathcal{N}(T), a \in \mathcal{A}(n)} \delta_2(T, n, a).$$

Although the algorithm described above is an efficient implementation of the optimal search, using dynamic programming, it is still quite expensive. To make the search practical for large vocabularies, it will probably be necessary to run a first pass HMM decoding to provide a word graph or lattice for rescoring with the segment model. This first pass need not use the same unit inventory as used by the segment model, though it is helpful if there are some hypothesized word and/or subword unit times by the first pass search to constrain the sets $\mathcal{N}(t)$ and further reduce computation. Other computational savings can be achieved by segment score caching and pruning. For insights into the relative savings possible, see [18].

5 Preliminary Experimental Results

To investigate the properties of the acoustic unit design algorithm, we conducted a number of experiments on the TIMIT database, using the male speakers from the New-England dialect-region (24 speakers for training, 7 for testing, 10 sentences per speaker). We generated 16 dimensional LPC derived cepstral feature vectors for the data using a 25.6 ms Hamming window with a 10 ms frame rate. The order of the LPC analysis was 24. A pre-emphasis coefficient of 0.97 was used in this parameterization. An energy coefficient was appended to the vectors. The increase of the average likelihood per frame on the training set, using an inventory size of 300 zeroth order full covariance models is depicted in figure 5. The increase of the average likelihood per frame on the training set, using an inventory size of 300 zeroth order full covariance models is depicted in figure 5, which suggests that only a few iterations of retraining are needed. The likelihood of iteration 0 is the likelihood of the models obtained after the initial clustering step.

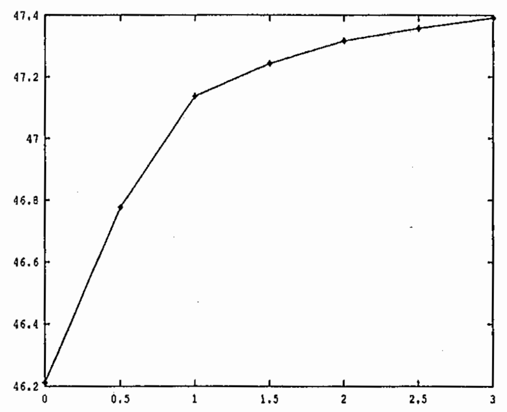

図 3: Likelihood as function of iterations on training data.

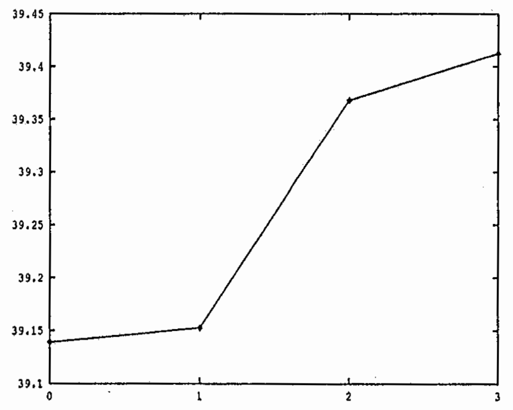

図 4: Likelihood as function of iterations on testing data.

表 1: Vowel classification results using different orders of the trajectory model, and full vs. diagonal covariances.

trajectory	diagonal covariance	full covariance
0-th order	47.86%	53.41%
1-st order	53.42%	56.26%

To compare the performance of the automatically-derived acoustic units versus phonetic units we constructed a set of allophonic unit models starting from the TIMIT phone segmentations. We separately clustered the segments corresponding to each phonetic label in the same way as for the acoustically-derived units described in section 2. The stopping criterion used for each cluster was that the average likelihood per frame was higher than some threshold P or that the number of frames was lower than some threshold M. We obtained 277 zeroth order allophonic unit models with diagonal covariance for 48 phone labels in this way. As a comparison, we computed an inventory of 277 acoustically-derived unit models with diagonal covariance and compared the likelihood, performing a Viterbi segmentation of the test set. The test set likelihood using the acoustically-derived unit models was higher than that based on the allophonic unit models given by the hand-labeled phones, which suggests that the acoustically-derived units fit the data better.

It has already been shown that higher order segment models outperform the zeroth order model in TIMIT vowel classification experiments [19], but we confirmed these results in our own vowel and phone classification experiments. In these experiments, we parameterized all the utterances in the TIMIT database using: 10 dimensional Mel-frequency scale cepstral coefficients computed from 24 filterbank outputs, a 5 ms. frame shift, a 25.6 ms Hamming window, a pre-emphasis coefficient of 0.97, and cepstral mean subtraction, where the cepstral means were computed from a whole utterance. In some of the experiments, we expanded this feature representation by adding delta and/or delta-delta parameters. The window size used for these computations was 2 frames.

Given this feature vector representation, we extracted the frames corresponding to a certain phone segment using the phone segment boundaries found in the label files of the TIMIT database and using the utterances from half of the speakers in the training set of TIMIT (231 speakers, 10 utterances per speaker). The distribution according to male and female speakers as well as the distribution according to dialect region was equal to the distribution found in the training set of the TIMIT database (i.e. every other speaker was selected). The full, 168 speaker TIMIT test set was used. These 168 speakers were not included in the training set. This resulted in a training set of 89166 phone segments and a test set of 62856 phone segments. Context independent (CI) models were constructed using the phone segments in the training set. For a 16 vowel experiment, we made CI model for the 16 TIMIT vowel labels. We also conducted a phone classification experiment using a model inventory of 48 phone labels (the Kai-Fu Lee phoneme set). The conversion to 48 labels is detailed in appendix 付録 A . All experiments use a relative frequency duration model.

For the 16 TIMIT vowel classification task using a feature representation of cepstral coefficient and delta coefficients, we obtained the results summarized in Table 1. As expected, the results show significant gains in performance from both types of increases in model complexity (first order trajectory and full covariance).

For the 48 phone classification task, the results are summarized in Tables 2 and 3. Again, note that using higher order models consistently improves classification performance. The gain from going from a zeroth order model to a first order model is greater than going from a first order model to a second order model. The phone labels that improve most by going from a zeroth order model to a first order model are the diphthongs. Another trend is that full covariance models consistently out perform diagonal covariance models. There is also a significant benefit to using delta cepstra, but performance drops when double derivatives are added, probably because the regression problem becomes ill-conditioned.

We tested the lexicon building algorithm by constructing a lexicon based on the automatic segmentations obtained by using 292 zeroth order models with diagonal covariance for all of the TIMIT corpus. The data included 326 male and 136 female speakers and 8 utterances per speaker, giving a total number of 4891

表 2: Phone classification results using a diagonal covariance, different orders of the trajectory model and different features.

trajectory	cepstra only	cepstra + Δcep	cepstra + $\Delta\Delta$cep
0-th order	31.73%	41.81%	37.73%
1-st order	40.29%	46.73%	43.14%
2-nd order	42.03%	48.30%	44.58%

表 3: Phone classification results using a full covariance, different orders of the trajectory model and different features.

trajectory	cepstra only	cepstra + Δcep	cepstra + $\Delta\Delta$cep
0-th order	36.16%	49.64%	47.65%
1-st order	43.36%	52.61%	50.30%
2-nd order	45.17%	53.66%	51.33%

distinct single word entries in a corpus of 30132 non-unique words. Using a 20 ms threshold for step 1 of the algorithm described in Section 3, 3422 distinct multi-word sequences were found. Multi-word sequences were pruned until a lexicon of size 5000 was obtained. The multi-word sequences were typically function word combinations, such as "in the", "do you", "of the", "on the", etc., as one might expect. Almost all additional lexical items were two word sequences. Many of the multi-word sequences that were not included were singleton examples of that word sequence.

6 Discussion

In this report, we have proposed a method for designing acoustically-derived segmental units, a lexicon containing multi-word entries for representing cross-word phonetic reduction/assimilation, and the mapping between a phonetic baseform associated with the lexicon and the observed segmental units. Preliminary experiments on the TIMIT corpus demonstrate the feasibility of each step, but it remains to be shown that the different components together offer a significant advantage over phonetic lexical modeling in continuous word recognition. However, we believe that the use of units that are potentially longer than phones and that can span word boundaries will be span word boundaries will be important for handling the reduction phenomena that occurs in spontaneous speech.

Two main issues are still to be addressed for obtaining good word recognition results. First, the model is unlikely to provide significant gains for piecewise constant segment means, but efficient search algorithms are needed for practical implementation of polynomial trajectories of higher order. Second, the simplifying assumptions made in the pronunciation model, equations 14 and 15, are too restrictive, and we plan to apply decision tree clustering techniques similar to those of [25] to enable conditioning on a broader context of phone units.

Acknowledgments

The contribution by Mari Ostendorf was sponsored in part by ATR and in part by ONR grant ONR-N00014-92-J-1778.

付録 A Phone Label Set Conversion

To convert the 61 TIMIT phone labels into this 48 phoneme set, the following rules were applied:

- The /q/ label is removed from the inventory.

- The /ux/ label is replaced with the uw label.
- The /axr/ label is replaced with the er label.
- The /em/ label is replaced with the m label.
- The /nx/ label is replaced with the n label.
- The /eng/ label is replaced with the ng label.
- The /hv/ label is replaced with the hh label.
- The /ax-h/ label is replaced with the ax label.
- The /pcl/, /tcl/, /kcl/ and /qcl/ labels are replaced with the cl label.
- The /bcl/, /dcl/ and /gcl/ labels are replaced with the vcl label.
- The h#, #h and pau labels are replaced with the /sil/ label.
- The j label is replaced with the jh label.

For the explanation of what these labels represent, see the documentation that comes with the TIMIT database.

参考文献

[1] C.-H. Lee, F. Soong and B.-H. Juang, "A segment model based approach to speech recognition," *Proc. of the Int. Conf. on Acoust., Speech and Signal Proc.*, pp. 501-504, 1988.

[2] L. Bahl, P. Brown, P. de Souza, R. Mercer and M. Picheny, "Acoustic Markov models used in the Tangora speech recognition system," *Proc. of the Int. Conf. on Acoust., Speech and Signal Proc.*, pp. 497-500, 1988.

[3] K. K. Paliwal, "Lexicon building methods for an acoustic sub-word based speech recognizer," *Proc. of the Int. Conf. on Acoust., Speech and Signal Proc.*, pp. 729-732, 1990.

[4] L. Bahl, P. de Souza, P. Gopalakrishnan, D. Nahamoo and M. Picheny, "Decision trees for phonological rules in continuous speech," *Proc. of the Int. Conf. on Acoust., Speech and Signal Proc.*, pp. 185-188, 1991.

[5] M.-Y. Hwang and X. Huang, "Subphonetic Modeling for Speech Recognition," in *Proc. DARPA Workshop on Speech and Natural Language*, 1992, pp. 174-179.

[6] V. Digalakis and H. Murveit, "Genones: Optimizing the degree of tying in a large vocabulary HMM-based speech recognizer," in *Proc. of the Int. Conf. on Acoust., Speech and Signal Proc.*, vol. I, 1994, pp. 537-540.

[7] J. Takami and S. Sagayama, "A successive state splitting algorithm for efficient allophone modeling," *Proc. of the Int. Conf. on Acoust., Speech and Signal Proc.*, vol. I, 1992, pp. 573-576.

[8] A. Kannan, M. Ostendorf and J. R. Rohlicek, "Maximum Likelihood Clustering of Gaussians for Speech Recognition," *IEEE Trans. on Speech and Audio Proc.*, vol. 2, no. 3, 1994, pp. 453-455.

[9] S. J. Young, J. J. Odell and P. C. Woodland, "Tree-based state tying for high accuracy acoustic modeling," in *Proc. ARPA Workshop on Human Language Technology*, 1994, pp. 307-312.

[10] D. Pallett *et al.*, "1994 Benchmark Tests for the ARPA Spoken Language Program," *Proc. ARPA Workshop on Spoken Language Technology*, pp. 5-38, 1995.

[11] T. Svendsen and F. Soong, "On the automatic segmentation of speech signals," *Proc. of the Int. Conf. on Acoust., Speech and Signal Proc.*, pp. 77-80, 1987.

[12] L. Bahl, J. Bellegarda, P. de Souza, P. Gopalakrishnan, D. Nahamoo, and M. Picheny, "A new class of fenonic Markov word models for large vocabulary continuous speech recognition," *Proc. of the Int. Conf. on Acoust., Speech and Signal Proc.*, pp. 177-180, 1991.

[13] S. Roucos, R. Schwartz and J. Makhoul, "Segment quantization for very low rate speech coding," *Proc. of the Int. Conf. on Acoust., Speech and Signal Proc.*, pp. 1565-1568, 1982.

[14] M. Ostendorf and S. Roukos, "A stochastic segment model for phoneme-based continuous speech recognition," *IEEE Trans. on Acoust., Speech, and Signal Proc.*, vol. 37, no. 12, pp. 1857-1869, 1989.

[15] Y. Linde, A. Buzo and R. M. Gray, "An algorithm for vector quantizer design," *IEEE Trans. Commun.*, vol. COM-28, pp. 84-95, Jan. 1980.

[16] Y. Shiraki and M. Honda, "LPC speech coding based on variable-length segment quantization," *IEEE Trans. on Acoust., Speech, and Signal Proc.*, vol. 36, no.9, pp. 1437-1444, 1988.

[17] P. Chou and T. Lookabaugh, "Variable dimension vector quantization of linear predictive coefficients of speech," *Proc. of the Int. Conf. on Acoust., Speech and Signal Proc.*, vol. I, pp. 505-508, 1994.

[18] M. Ostendorf, V. Digalakis and O. Kimball, "From HMMs to Segment Models: A Unified View of Stochastic Modeling for Speech Recognition," Boston University ECS Department Technical Report NO. ECS-95002, 1995.

[19] H. Gish and K. Ng, "A segmental speech model with applications to word spotting," in *Proc. of the Int. Conf. on Acoust., Speech and Signal Proc.*, 1993, pp. II-447-450.

[20] L. Deng, M. Aksmanovic, D. Sun and J. Wu, "Speech recognition using hidden Markov models with polynomial regression functions as nonstationary states," *IEEE Trans. on Speech and Audio Proc.*, vol. 2, no. 4, pp. 507-520, 1994.

[21] A. Asadi, R. Schwartz and J. Makhoul, "Automatic modeling for adding new words to a large vocabulary speech recognition system," *Proc. of the Int. Conf. on Acoust., Speech and Signal Proc.*, pp. 305-308, 1991.

[22] J. M. Lucassen and R. L. Mercer, "An information theoretic approach to the automatic determination of phonemic baseforms," *Proc. of the Int. Conf. on Acoust., Speech and Signal Proc.*, vol. III, pp. 42.5.1-42.5.4, 1984.

[23] M. Cohen, "Phonological structures for speech recognition," Ph.D. thesis, University of California, Berkeley, 1989.

[24] A. Ljolje, M. Riley, D. Hindle and F. Perreira, "The AT&T 60,000 word speech-to-text system," *Proc. ARPA Workshop on Spoken Language Technology*, pp. 162-165, 1995.

[25] M. Riley, "A statistical model for generating pronunciation networks," in *Proc. of the Int. Conf. on Acoust., Speech and Signal Proc.*, vol. II, pp. S11.1-S11.4, 1991.

[26] E. Sanchis, F. Casacuberta, I. Galiano and E. Segarra, "Learning structural models of subword units through grammatical inference techniques," *Proc. of the Int. Conf. on Acoust., Speech and Signal Proc.*, pp. 189-192, 1991.

[27] C. Wooters and A. Stolcke, "Multiple-pronunciation lexical modeling in a speaker independent speech understanding system," *Proc. Int'l. Conf. on Spoken Language Proc.*, pp. 1363-1366.

[28] R. De Mori, C. Snow and M. Galler, "On the use of stochastic inference networks for representing multiple word pronunciations," *Proc. of the Int. Conf. on Acoust., Speech and Signal Proc.*, pp. 568-571, 1995.

[29] Y.-L. Chow and R. Schwartz, "The N-Best Algorithm: An Efficient Procedure for Finding Top N Sentence Hypotheses," *Proceedings of the Second DARPA Workshop on Speech and Natural Language*, pp. 199–202, October 1989.

[30] L. R. Bahl *et al.*, "Automatic phonetic baseform determination," *Proc. of the Int. Conf. on Acoust., Speech and Signal Proc.*, pp. 173-176, 1991.

[31] H. Meng, S. Seneff and V. Zue, "Phonological parsing for reversible letter-to-sound/sound-to-letter generation," *Proc. of the Int. Conf. on Acoust., Speech and Signal Proc.*, vol. II, pp. 1-4, 1994.