FEATURE PRUNING IN LIKELIHOOD EVALUATION OF HMM-BASED SPEECH RECOGNITION

Xiao Li and Jeff Bilmes

Department of Electrical Engineering
University of Washington, Seattle
{lixiao, bilmes}@ee.washington.edu

ABSTRACT

In this work, we present a simple yet effective technique to reduce the likelihood computation in ASR systems that use continuous density HMMs. In a variety of speech recognition tasks, likelihood evaluation accounts for a significant portion of the total computational load. Our proposed method, under certain conditions, only evaluates the component likelihoods of certain features, and approximates those of the remaining (pruned) features by prediction. We investigate two feature clustering approaches associated with our pruning technique. While a simple sequential clustering works remarkably well, a data-driven approach performs even better in its attempt to save computation while maintaining baseline recognition accuracy. With the second approach, we can speed up the likelihood evaluation by 33% and reduce its power consumption by 27% for an isolated word recognition task. For a continuous speech recognition system using either monophone or triphone models, the speedup and power reduction of the likelihood evaluation are 50% and 35% respectively.

1. INTRODUCTION

The proliferation of mobile devices has caused a great demand for a more convenient user interface, and speech technology is widely believed to be an ideal methodology for this purpose. Portable devices, however, usually do not have abundant resources to support the heavy computational load of a speech recognition engine. While computation time keeps decreasing with the development of modern microprocessors, power consumption remains an impediment to the ubiquitous adoption of speech technology for portable devices.

For ASR systems using continuous HMMs, the state likelihood evaluation can dominate the operational load by taking up to 96% of the total computation for a typical small vocabulary application [1], and 30% to 70% of the total computation for LVCSR tasks [2]. Since the observation distribution is typically represented by a Gaussian mixture, the computational complexity of the likelihood evaluation is proportional to the total number of Gaussians in the system and the dimensionality of these Gaussians. This gives two potential research directions to improve efficiency: reducing the number of Gaussians and reducing their dimensionality.

The number of Gaussian evaluations can be reduced in a system that employs state tying since, once evaluated, the Gaussian score can be stored and utilized multiple times. Tying has been extensively applied to different levels of the hierarchical structure of HMMs [3, 4, 5, 6], and tends to be implemented in even finer units such as subspace distributions [7, 8]. Alternately, Gaussians can be pruned online using Gaussian selection [1, 2], where only a fraction of Gaussians with means close to the observation are precisely computed for each state, and all remaining ones are assigned approximated values. Taking the other direction, various feature selection methods [9, 10, 11] tackle the dimensionality problem by selecting only the features with the highest discriminative power. Also, there has been work related to our method [12], where incremental feature pruning was applied to discrete-mixture models.

In this paper, we achieve a reduction in likelihood computation and its power consumption by selectively computing likelihoods of feature subsets. Specifically, we precisely compute the likelihoods of certain feature subsets first, and conditionally approximate those of other (pruned) subsets by prediction. The amount of pruning depends on how the features are clustered and ordered. We describe the pruning algorithm in detail in section 2, and present two associated feature clustering schemes in section 3. In section 4, we show that with data-driven clustering, we can speed up the likelihood evaluation by 33% to 50% with only a slight degradation in system accuracy, though the overall speedups depend on the ratio of search to likelihood evaluation. In addition, power simulations show a reduction of 27% to 35% in terms of power consumed by the likelihood evaluation. As will be seen, independent of the training process, our technique is easily incorporated into any standard decoder.

This work was funded by NSF grant ITR/RC-0086032.

2. FEATURE PRUNING

In continuous HMMs, the density $b_j(O)$ of the observation vector O given a certain state j is typically parameterized by a Gaussian mixture,

$$b_j(O) = \sum_{i \in M_j} w_i \mathcal{N}(O; \mu_i, \Sigma_i) \qquad (1)$$

where $O = (x(1), x(2), ..., x(D))$ with D the dimensionality of the features and M_j the subset of Gaussians which belongs to state j.

If we assume diagonal covariance matrices, the log probability of a single D-dimensional Gaussian is computed as,

$$\ln \mathcal{N}(O; \mu_i, \Sigma_i) = -\frac{c_{0i}}{2} - \sum_{j=1}^{D} \frac{(x(j) - \mu_i(j))^2}{2\sigma_i^2(j)} \qquad (2)$$

where c_{0i} is a constant independent of the observation, which can be computed offline. The online computation of the Mahalanobis distance for each dimension is usually performed sequentially.

In a diagonal covariance mixture of Gaussian HMM, it is implicitly assumed that feature subsets are conditionally independent given the HMM state and the mixture component identity. This property allows the Gaussian over the full-space to be decomposed into K lower-dimensional Gaussians over orthogonal subspaces each with dimension d_k, k=1..K, and $\sum_{k=1}^{K} d_k = D$. We let $q_{ik}(O)$ denote the k^{th} subspace's negative log Gaussian probability. Note that $q_{ik}(O)$ in fact uses only a subset of the features within O, and the features used need not be contiguous. The i^{th} component likelihood then becomes

$$\ln \mathcal{N}(O; \mu_i, \Sigma_i) = -\sum_{k=1}^{K} q_{ik}(O) \qquad (3)$$

with $q_{ik}(O)$ expressed as

$$q_{ik}(O) = \frac{c_{0ik}}{2} + \sum_{j=1}^{d_k} \frac{(x_k(j) - \mu_{ik}(j))^2}{2\sigma_{ik}^2(j)} \qquad (4)$$

The Gaussian mixture model has some interesting numerical properties. If $q_{i1}(O)$ in (3) is small (meaning the corresponding subspace Gaussian has very high probability), $q_{ik}(O)$, $k = 2..K$, would become dominant in determining $\mathcal{N}(O; \mu_i, \Sigma_i)$. If, on the other hand, $q_{i1}(O)$ is large, then $\mathcal{N}(O; \mu_i, \Sigma_i)$ would be no higher than $\exp\{-q_{i1}(O)\}$. The i^{th} component, as suggested by (1), would have a contribution of at most $w_i \exp\{-q_{i1}(O)\}$ to the state likelihood $b_j(O)$. That contribution quickly becomes negligible when $q_{i1}(O)$ gets higher than a threshold, where computing $q_{ik}(O)$, k=2..K would not provide much additional resolution in the likelihood value.

Therefore, we can achieve a saving by conditional approximations to the likelihood evaluation. We only compute $q_{ik}(O)$, k=2..K, when $q_{i1}(O)$ is small enough. Otherwise, we ignore the expensive remaining Gaussian probability computation and assign approximate values. In fact, the comparison and approximation can be performed at multiple levels. In this work, we let the indices $1..K$ indicate the order in which the subspace likelihoods are computed. The feature pruning algorithm can be describe as follows,

Input: observation O; Gaussian parameters $\{\mu_i, \Sigma_i\}$
Output: approximated $\ln \mathcal{N}(O; \mu_i, \Sigma_i)$

```
1:  $a \leftarrow q_{i1}(O)$;
2:  for $k = 2..K$ do
3:      if $a < t_{k-1}$ then
4:          compute $q_{ik}(O)$ according to (4);
5:          $a \leftarrow a + q_{ik}(O)$;
6:      else
7:          $a \leftarrow f_{k-1}(a)$;
8:          break;
9:      end if
10: end for
11: return $-a$;
```

where t_k, $k = 1..K$-1, are a set of thresholds, and $f_k(\cdot)$ are a set of affine approximation functions whose coefficients can be determined using Least Mean Square (LMS) estimation on the training data. Consequently, only a fraction of observation vectors are precisely computed to full dimension. Some of the features are ignored according to the likelihoods of the already-computed features.

The tradeoff between recognition accuracy and computation time or power consumption is controlled by the thresholds, the approximation functions, and the way that the features are clustered into subspaces and are ordered. First, the thresholds control both approximation accuracy and computation/power by determining how often approximation takes place. Second, the approximation functions influence the accuracy by their overall fidelity, but do not affect computation/power. Third, the feature clustering scheme controls accuracy by its success in selecting feature subsets that have high correlation in likelihood (thereby making prediction easy), and also controls computation/power via the size of the subsets chosen (which determine the degree of pruning when it occurs).

3. CLUSTERING OF FEATURES

Ideally the features should be clustered in such a way that we can prune as much as possible while maintaining baseline recognition rate. In this section, we present two clustering schemes: the first is purely sequential and the second uses a data-driven approach.

3.1. Sequential clustering

For recognizers which use both static and dynamic features, a simple and intuitive way to cluster features is sequential clustering. More specifically, the static features are allocated to the first group, their deltas to the second, and the double deltas to the last if available. The advantage is that the features are very likely to be represented in this order, which simplifies the incorporation of feature pruning into an existing ASR system. Furthermore, as is explained in the following text, we observe a certain degree of correlation between the component likelihoods of the dynamic features and those of the static features. And the predictability of the likelihoods is beneficial to our feature pruning technique.

Dynamic features in conjunction with static features have long been used in ASR systems. [13] gives a discriminative explanation of why dynamic features can help in speech recognition in spite of the fact they are generated merely from static features. However, such a concatenated feature vector inevitably contains redundant information and may introduce a certain degree of correlation between the likelihoods of the static and dynamic features. Furthermore, dynamic features improve the recognition rate but at the cost of doubling or even tripling the computational effort in the likelihood evaluation. Therefore, it should at least be questioned whether the improvement in performance is worth the increase in computational cost.

To illustrate the correlation between the subspace likelihoods of static and dynamic features, we observe the joint distribution of $q_{i1}(O)$ and $q_{i2}(O)$ of certain phonemes. Note again that large $q_{i1}(O)$ and $q_{i2}(O)$ indicate low likelihoods. Assuming $q_{i1}(O)$ and $q_{i2}(O)$ were independent random variables, if O was itself a Gaussian, the value of $q_{i1}(O)$ and $q_{i2}(O)$ would each have a χ-square distribution. As a simple demonstration, we draw from two identical and independent χ-square random variables, as depicted in the upper-left plot (Pattern A) of figure 1, where we are unable to obtain any information about $q_{i1}(O)$ from $q_{i2}(O)$. On the other hand, if they are fully correlated as in the upper-right plot (Pattern B), we can estimate the former likelihood completely from the latter, and the dynamic features are absolutely redundant in this sense. The real joint distribution, however, lies somewhere between the two above extreme cases. The bottom plots of figure 1 show the true $q_{i1}(O)$ and $q_{i2}(O)$, $\forall i$, with observation O chosen randomly from various utterances from the TIMIT corpus. The bottom-left figure plots the data distribution for all states of the vowel /i/ and the bottom-right one for the consonant /C/. As shown in the figure, the conditional variance VAR$[q_{i2}(O)|q_{i1}(O)]$ tends to decrease as $q_{i1}(O)$ increases. In other words, if the component likelihood of the static features is low, the likelihood of the dynamic features has a relatively narrow dynamic range, which allows us to more accurately approximate the latter based on the value of the former. The plots for other phonemes follow the same trend, and we observe similar patterns for $q_{i3}(O)$ vs. $q_{i2}(O)$ when double deltas are used.

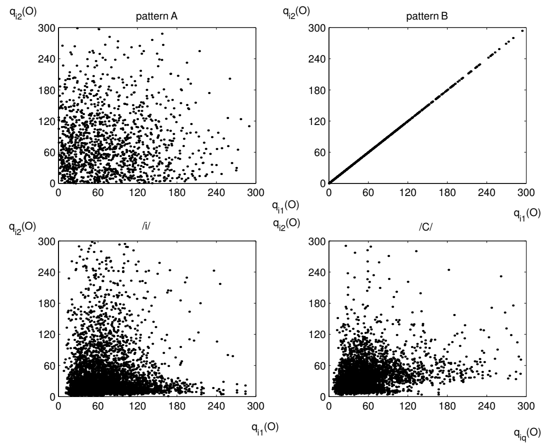

Figure 1: $q_{i2}(O)$ vs. $q_{i1}(O)$. Upper: two extreme distribution patterns; Bottom: two real distributions for vowel and consonant phonemes respectively

To determine the affine functions $f_k(\cdot)$, we assume $f_k(t) = r_k t + s_k$ and apply LMS estimation on a subset of the training data where $\sum_{l=1}^{k} q_{il}(O) > t_k$. Experiments show that $r_k \approx 1$, and we therefore let $r_k = 1$ for simplicity. In other words, in the pruning algorithm whenever $a > t_k$, we let $a \leftarrow a + s_k$ and return $-a$ as the approximated component likelihood.

3.2. Data-driven clustering

For more general features, we do not have knowledge in advance about the correlation between their component likelihoods. The complexity of trying every combination to find the best clustering scheme is factorial in the dimension of the features. Therefore, we evaluate a data-driven method to cluster the features according to their statistical properties. In this work, we only study the case where the features are grouped into clusters with the same size.

First, we find from empirical experimentation that the recognition rate depends heavily on the thresholds t_k, $k = 1..K$-1, no matter how the features are clustered. In other words, given K and a set of t_k, we tend to get similar recognition rates for all possible clustering schemes. Secondly, the clustering does have an impact on the recognition rate, but at a smaller scale, since it more or less affects the predictability of the likelihoods and hence the effectiveness of the approximation functions. Finally, the clustering has a great impact on the savings in computation and power; different clustering schemes would have different numbers of feature clusters pruned under the same thresholding value. Therefore, since the thresholds very much decide the recognition rate, we hope that the clustering will prune as many features as possible before the critical point where the recognition rate begins to drop from the baseline. If t_1 is the critical threshold point, the first group of features to be computed should be selected to have the maximum probability of having a score below the threshold, thereby increasing the chance that the remaining feature likelihoods will not need to be computed. Specifically, we search for a set of features that attempts to maximize $Pr\{q_{i1}(O) > t_1\}$, which can be estimated non-parametrically using a large amount of data. While the optimal clustering is intractable, we instead employ the following greedy algorithm:

1. Determine the number of clusters K according to the total number of features D. (Note the optimal choice of K is beyond the scope of our work. There were experiments related to this topic in [12].) For comparison, we set K the same as that used in sequential clustering, namely 2 or 3, in our work.

2. Randomly cluster the features into K groups with the same size, perform feature pruning experiments with decreasing t_1 ($t_2..t_{K-1}$ are decreased accordingly), and find the critical point $t_1 = t_0$. Repeat the step for different random clusterings to get multiple t_0's, and we simply use the mean, $\text{E}[t_0]$.

3. Extract a subset of representative training data and compute the log component likelihoods over all states in each feature dimension. For each of the D features, count the number of log likelihoods below $\text{E}[t_0]/D$. List the features in descending order by that number.

4. Sequentially cluster the features into K groups according to the new order.

The essential idea of the clustering is that we assign a higher priority to features with a larger number of low component likelihoods. Here we mean "low" by comparing it with a certain threshold.

Note that in the clustering algorithm, we do not take into consideration the correlation between likelihoods. Experiments show, however, the conditional variances

$$\text{VAR}[q_{i(k+1)}(O)|q_{ik}(O) > t_k]$$

are smaller than those obtained from a random clustering. In other words, this scheme gives a relatively high predictability in likelihoods. Again, we apply LMS estimation to learn the approximation functions.

4. EXPERIMENTS AND RESULTS

We ran two sets of experiments, one on an isolated word corpus and the other on a continuous speech recognition task. The speedup and power reduction of the likelihood evaluation were impressive for both tasks.[1]

[1]In this work, we define

$$\%\text{ speedup} = \left(\frac{\text{baseline CPU time}}{\text{new CPU time}} - 1\right) \times 100\%$$

$$\%\text{ power reduction} = \left(1 - \frac{\text{new CPU power}}{\text{baseline CPU power}}\right) \times 100\%$$

4.1. Isolated word recognition on PhoneBook

The isolated word experiment was done on NYNEX Phonebook [14], a telephone-speech database. We used 42 continuous HMMs as an acoustic model with 165 states altogether. Each state was represented by a Gaussian mixture comprised of 12 Gaussian components. We extracted MFCCs, log energy and their deltas, leading to 26 features. The test was carried out on 8 different testing sets with 75 words each. The final WER was an average over them all. The baseline WER was 2.07%, and the likelihood evaluation was responsible for 70% of the total computation without beam pruning and over 80% when beam search was applied.

We clustered the features into two groups, where only one threshold t_1 and one approximation function $f_2(\cdot)$ were needed for feature pruning. The WER began to rise when t_1 was decreased to the critical point. For each threshold, we got the WER in parallel with the speedup of the likelihood evaluation.

1st group	2nd group
e	c_7
Δe	c_8
c_1	c_9
c_2	c_{10}
c_3	c_{11}
c_4	c_{12}
c_5	Δc_6
c_6	Δc_7
Δc_1	Δc_8
Δc_2	Δc_9
Δc_3	Δc_{10}
Δc_4	Δc_{11}
Δc_5	Δc_{12}

Table 1. Data-driven feature clustering for PhoneBook. (e: log energy; c: MFCCs; Δ: deltas)

We ran feature pruning experiments using our proposed clustering schemes. With sequential clustering, all the MFCCs were in the first group and their deltas in the second. We also applied data-driven clustering with two equally sized groups, as shown in table 1. Interestingly, the clustering displayed certain patterns. For example, the log energy and the first half of the MFCCs along with their deltas were allocated to the first group. This clustering indicates to some extent which features are most important in the likelihood evaluation. For comparison, we also generated randomly clustered groups from the original 26 features, with 13 features in each group.

From analyzing our experimental results, we found that the critical threshold points for all three of these clustering schemes agree very well, where in each case we observe around a 0.2% absolute increase in WER. In terms of speedups, however, the data-driven scheme is better than the other two. As shown in figure 2, the sequential clustering achieves a speedup by above 20% in the likelihood evaluation. The data-driven approach, however, outperforms it by achieving a speedup of 33%. The random clustering can still speed up the computation by 12%, although the results of different random clusterings vary insignificantly.

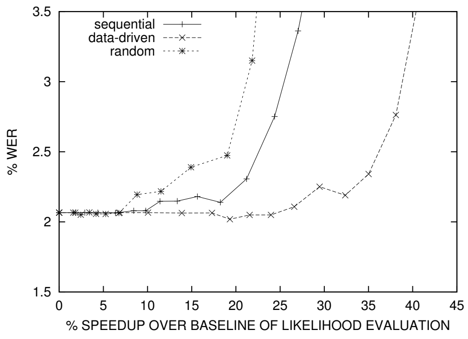

Figure 2: Feature pruning on PhoneBook

4.2. Continuous speech recognition on TIMIT

Our continuous speech recognition system was based on TIMIT. We applied the pruning technique to both monophone and triphone systems. In the monophone system, we used a simple bi-gram language model and 42 monophone HMMs for the acoustic model. We assigned 3 emitting states to each HMM which amounted to 126 states altogether. Each state distribution was represented by a Gaussian mixture with 12 components. The features we used were MFCCs, log energy and their delta and double deltas, leading to 39 dimensions. The baseline computes each observation vector to its full dimension for all 1512 Gaussian components. As for the triphone system, states were tied into 1356 distinct ones using a decision tree. Each state distribution had 6 Gaussians which resulted in a system with 8135 Gaussian components. For both systems, one-pass decoding was performed on the complete TIMIT testing set consisting of 1344 utterances. The baseline accuracy was 85.68% for the monophone system, and 88.20% for the triphone system.

We divided the features into three equally sized groups. We started with very high thresholds t_1, t_2, and gradually decreased them simultaneously. Note that we always obeyed $t_2 = t_1 + t'$, and the specific value of t' was chosen empirically which gave the best result.

We again evaluated both proposed clustering schemes. In sequential clustering, we observed the correlation between the likelihoods of static features, deltas, and double deltas, and made the approximation by adding offsets s_1, s_2, as stated in section 3. Remarkably, in the experiments using data-driven clustering, the monophone and triphone systems produced the same clustering (modulo a permutation) which is shown in table 2, and has a very similar pattern both to table 1 and to [10]. The log energy along with its delta and double deltas were grouped together to be computed first; most of the MFCCs were assigned a higher computational priority than their delta and double delta counterparts. Again, we included a random clustering for comparison.

1st group	2nd group	3rd group
e	c_7	c_{11}
Δe	c_8	c_{12}
$\Delta^2 e$	c_9	Δc_7
c_1	c_{10}	Δc_8
c_2	Δc_3	Δc_9
c_3	Δc_4	Δc_{10}
c_4	Δc_5	Δc_{11}
c_5	Δc_6	Δc_{12}
c_6	$\Delta^2 c_3$	$\Delta^2 c_7$
Δc_1	$\Delta^2 c_4$	$\Delta^2 c_9$
Δc_2	$\Delta^2 c_5$	$\Delta^2 c_{10}$
$\Delta^2 c_1$	$\Delta^2 c_6$	$\Delta^2 c_{11}$
$\Delta^2 c_2$	$\Delta^2 c_8$	$\Delta^2 c_{12}$

Table 2. Data-driven feature clustering for TIMIT. (e: log energy; c: MFCCs; Δ: deltas; Δ^2: double deltas)

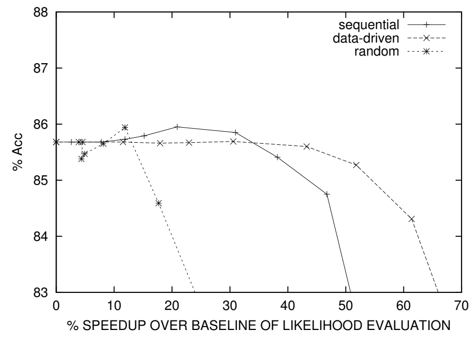

Figure 3: Feature pruning on TIMIT monophone system

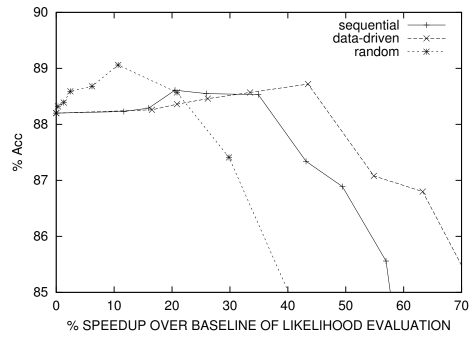

Figure 4: Feature pruning on TIMIT triphone system

As shown in figure 3, for the monophone system, the sequential clustering can speed up the likelihood evaluation by 40% with a 0.2% absolute decrease in accuracy. The data-driven approach works even better with a speedup over 50%. A random clustering is obviously much worse compared with our clustering techniques. Similar results are observed in figure 4 for the triphone system, where our clustering schemes appear to be very effective in reducing likelihood computation with little loss in accuracy.

4.3. Power simulation

To estimate the power consumed by the likelihood evaluation, we utilized Wattch, a framework for architecture-level power dissipation analysis [15]. We compiled the baseline system and the systems integrated with different amount of feature pruning, all targeted for PISA architecture [16]. Power simulation was then launched and cycle-level resource expenses were obtained, though only the power dissipation of likelihood evaluation was extracted for our interest. Table 3 shows the percentage of power reduction in the likelihood evaluation for different clustering schemes on different tasks, where feature pruning with data-driven clustering results in a reduction of 27% to 35%. The overall reduction again depends on the ratio of likelihood evaluation to search and the other factors such as vocabulary size.

Clustering	PhoneBook	TIMIT mono.	TIMIT tri.
random	16%	12%	21%
sequential	21%	30%	29%
data-driven	27%	35%	34%

Table 3. Power savings in likelihood evaluation

5. CONCLUSION AND DISCUSSION

In comparison with conventional efficient decoding techniques, our approach focuses on reducing likelihood computation and power consumption by partially computing the observation probability in a Gaussian component. It ignores computing the remaining (pruned) part of an observation vector when the observation gets a low probability in a subspace Gaussian. We achieved a speedup of 33% to 50% and a power reduction of 27% to 35% in the likelihood evaluation of various recognition tasks. In addition, our technique does not require a complicated training procedure and is complementary to other fast search techniques, such as Gaussian selection and beam search. It is noticeable that there are many variations on our technique; the number of feature clusters can be optimized, and the thresholds and offsets can be customized for different phonemes or HMM states.

We thank Jonathan Malkin for reading early drafts of this work.

6. REFERENCES

[1] E. Bocchieri, "Vector quantization for the efficient computation of continuous density likelihoods," in *Proc. ICASSP*, 1993, vol. 2, pp. 692–695.

[2] M.J.F.Gales, K.M.Knill, and S.J.Young, "State based gaussian selection in large vocabulary continuous speech recognition using hmms," *IEEE Trans. on Speech and Audio Processing*, vol. 7, 1999.

[3] K.F.Lee, S.Hayamizu, H.W.Hon, C.Huang, J.Swartz, and R.Weide, "Allophone clustering for continuous speech recognition," in *Proc. ICASSP*, 1990, vol. 2, pp. 749–752.

[4] S.J.Young and P.C.Woodland, "The use of state tying in continuous speech recognition," in *Proc. Eurospeech*, 1993, pp. 2203–2206.

[5] X.D. Huang and M.A. Jack, "Semi-continuous hidden markov models in isolated word recognition," in *9th International Conference on Pattern Recognition*, November 1988, vol. 1, pp. 406–408.

[6] J.R.Bellegarda and D.Nahamoo, "Tied mixture continuous parameter modeling for speech recognition," *IEEE Trans. on Acoustics, Speech and Signal Processing*, vol. 38, pp. 2033–2045, 1990.

[7] S.Takahashi and S.Sagayama, "Four-level tied-structure for efficient representation of acoustic modeling," in *Proc. ICASSP*, 1995, vol. 1, pp. 520–523.

[8] E.Bocchieri and B.K.Mak, "Subspace distribution clustering hidden markov model," *IEEE Trans. on Speech and Audio Processing*, vol. 9, no. 3, pp. 264–275, 2001.

[9] E. Lleida and C. Nadeu, "Principal and discriminant component analysis for feature selection in isolated word recognition," in *Signal Processing V: Theories and Applications*, pp. 1251–1254. Elsevier Sc. Publ. B.V., 1990.

[10] E. Bocchieri and J. Wilpon, "Discriminative feature selection for speech recognition," *Computer, Speech and Language*, pp. 229–246, 1993.

[11] J. Nouza, "Feature selection methods for hidden markov model-based speech recognition," in *Proc. Intl. Conf. on Pattern Recognition*, 1996, vol. 2, pp. 186–190.

[12] V.Digalakis, S.Tsakalidis, C.Harizakis, and L.Neumeyer, "Efficient speech recognition using subvector quantization and discrete-mixture hmms," *Computer Speech and Language*, vol. 14, pp. 33–46, 2000.

[13] J. A. Bilmes, "Graphical models and automatic speech recognition," in *Mathematical Foundations of Speech and Language Processing*, R. Rosenfeld, M. Ostendorf, S. Khudanpur, and M. Johnson, Eds. Springer-Verlag, New York, 2003.

[14] J.F.Pitrelli, C.Fong, and H.C.Leung, "Phonebook: A phonetically-rich isolated-word telephone-speech database," in *Proc. ICASSP*, 1995.

[15] D.Brooks, V.Tiwari, and D.Martonos, "Wattch: A framework for architecturelevel power analysis and optimizations," in *Proc. Intl. Symp. on Computer Architecture*, 2000, pp. 83–94.

[16] D.Burger and T.M.Austin, "The SimpleScalar tool set, Version 2.0," Tech. Rep. 1342, Univ. of Wisconsin-Madison, 1997.