Exploiting Correlations Among Models with Application to Large Vocabulary Speech Recognition

Ronald Rosenfeld Xuedong Huang Merrick Furst

May 1991
CMU-CS-91-148

School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213

Abstract

In a typical speech recognition system, computing the match between an incoming acoustic string and many competing models is computationally expensive. Once the highest ranking models are identified, all other match scores are discarded. We propose to make use of all computed scores by means of statistical inference. We view the match between an incoming acoustic string s and a model M_i as a random variable Y_i. The class-conditional distributions of $(Y_1, \ldots, Y_N)$ can be studied offline by sampling, and then used in a variety of ways. For example, the means of these distributions give rise to a natural measure of distance between models.

One of the most useful applications of these distributions is as a basis for a new Bayesian classifier. The latter can be used to significantly reduce search effort in large vocabularies, and to quickly obtain a short list of candidate words. An example HMM-based system shows promising results.

This research was supported by the Defense Advanced Research Projects Agency and monitored by the Space and Naval Warfare Systems Command under Contract N00039-91-C-0158, ARPA Order No. 7239.

The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of DARPA or the U.S. government.

Keywords: artificial intelligence, speech recognition, hidden markov models, fast match, large vocabulary.

1. Motivation and Outline

During the recognition phase of a typical speech recognition system, an incoming speech segment s is matched against a large number of competing models $M_1, M_2, \ldots, M_N$. The model or models that score the highest are then selected for further consideration.

There are many variations on this basic idea. The speech model may represent a phoneme, a syllable or a word. The competing models may represent the entire vocabulary, or only that part of it allowed by the grammar. The type of models used may vary, together with the matching process. Template-based models would typically be used with dynamic time warping ([Sakoe & Chiba 78]) and some metric distance defined over frames. In HMM-based systems, a match is typically defined as the class-conditional log-probability ($\log P(s|M_i)$). The models themselves may be trained as Maximum Likelihood estimators, or else optimized for discrimination ([Bahl et al. 88a]).

Common to all of these scenarios, however, are the following:

1. Computing the match for all the models is computationally expensive. For large vacabularies, it is prohibitive.
2. Once the best scoring models have been identified, all other match scores are discarded.

These observations suggest that a lot of computation is wasted in this process.

The main idea of this work is to remedy this situation by making use of all the computed scores. This can be done using statistical inference techniques. In order to be able to use all scores at run time, though, we must first analyze offline the statistical relationships between the models.

In section 2, we present a framework for such an analysis. Section 3 demonstrates the usefulness of this framework, by showing how it naturally gives rise to a measure of distance between models. In section 4 we develop the main application, namely reducing search time in large vocabularies. Finally, in section 5 we speculate about other possible uses for this framework.

2. Framework

Let

$$Y_i(s) = \text{the match between acoustic string } s \text{ and model } M_i. \tag{1}$$

$Y_i(s)$ could be any reasonable measure of agreement between a model and an acoustic instance.

Consider the Y_i's as random variables. The distribution of $\mathbf{Y} \stackrel{\text{def}}{=} (Y_1, Y_2, \ldots Y_N)$ is determined by the population from which the s's are taken. Let

$$D_j(\mathbf{Y}) \stackrel{\text{def}}{=} P(\mathbf{Y}|s \in \text{speech-unit-}j). \tag{2}$$

	$Y_1(s)$	$Y_2(s)$	...	...	...	$Y_i(s)$	...	...	...	$Y_N(s)$
D_1	$D_1(Y_1)$	$D_1(Y_2)$	...	...	...	$D_1(Y_i)$	...	...	...	$D_1(Y_N)$
D_2	$D_2(Y_1)$	$D_2(Y_2)$	...	...	...	$D_2(Y_i)$	...	...	...	$D_2(Y_N)$
⋮	⋮	⋮				⋮				⋮
⋮	⋮	⋮				⋮				⋮
⋮	⋮	⋮				⋮				⋮
D_j	$D_j(Y_1)$	$D_j(Y_2)$	...	...	...	$D_j(Y_i)$	...	...	...	$D_j(Y_N)$
⋮	⋮	⋮				⋮				⋮
⋮	⋮	⋮				⋮				⋮
⋮	⋮	⋮				⋮				⋮
D_N	$D_N(Y_1)$	$D_N(Y_2)$	...	...	...	$D_N(Y_i)$	...	...	...	$D_N(Y_N)$

Figure 1: The N^2 distributions $D_j(Y_i) \stackrel{\text{def}}{=} P(Y_i|s \in \text{speech-unit-}j)$. A row represents distributions of different Y values based on the same population of acoustic strings. A column represents distributions of the same Y value under different string populations.

$D_j(\mathbf{Y})$ is an N-dimensional distribution. It can be estimated by selecting examples of speech unit j and evaluating them by all the models. This can be done for all N distributions $D_1, D_2, \ldots D_N$.

We may also wish to consider the univariate distribution of each individual Y_i. We define:

$$D_j(Y_i) \stackrel{\text{def}}{=} P(Y_i|s \in \text{speech-unit-}j). \tag{3}$$

There are N^2 such distributions, which can be estimated in a similar way. Figure 1 depicts these distributions symbolically. Row j represents the distribution of different Y values under strings s from population j. Column i represents the distributions of Y_i under different string populations.

For generative models, if $M_1, M_2, \ldots, M_N$ are good models of speech units $1, 2, \ldots, N$, respectively, then D_j can be approximated by D_j^*, where

$$D_j^*(\mathbf{Y}) \stackrel{\text{def}}{=} P(\mathbf{Y}|M_j), \tag{4}$$

and similarly for $D_j^*(Y_i)$. $D_1^*, D_2^*, \ldots, D_N^*$ can be estimated in the same way as $D_1, D_2, \ldots, D_N$, except that the strings s are now generated by models $M_1, M_2, \ldots, M_N$, respectively.

These definitions, and the following analysis, apply to models of any type. For empirical support, we chose to apply these ideas to a small SCHMM-based system ([Huang et al. 90]) of 48 context-independent English phoneme models, as used in the baseline SPHINX system ([Lee 88]). We chose SCHMM over the discrete model because distance between acoustic strings is better modeled in Continuous HMM or SCHMM (since they are not subject to VQ errors). For this system we define:

$$Y_i(s) = \frac{1}{|s|} \log P(s|M_i) \tag{5}$$

	/ae/	/ax/	/ay/	/w/	/ng/	/g/	/sh/	/dd/
/ae/	7.68	5.39	4.53	-1.23	1.28	-1.72	-2.66	0.23
/ax/	-5.25	0.68	-10.35	-6.49	-5.15	-6.94	-10.24	-5.33
/ay/	6.76	5.13	8.95	0.13	0.80	-1.08	-2.36	-0.38
/w/	-9.55	-4.08	-10.45	1.55	-7.11	-5.99	-12.09	-6.48
/ng/	-3.70	-0.23	-5.93	-3.62	4.25	-1.81	-6.13	-0.37
/g/	-13.96	-9.20	-15.34	-8.71	-9.98	-0.44	-10.73	-2.60
/sh/	-5.53	-3.50	-6.00	-4.91	-3.60	-0.87	7.80	0.21
/dd/	-15.39	-12.41	-16.59	-13.40	-11.28	-8.59	-13.29	-0.19

Table 1: A submatrix of $E^* \stackrel{\text{def}}{=} E^*_{ji}$ (the means of the $D^*_j(Y_i)$'s). The diagonal entries are the row maxima but not necessarily the column maxima. See the text.

3. Example: Deriving a Measure of Distance Between Models

To illustrate the usefulness of our formalism, we now use it to derive a natural measure of distance between models.

Consider the means of the $D_j(Y_i)$'s:

$$E_{ji} \stackrel{\text{def}}{=} E[D_j(Y_i)] = \int P(s|\text{speech-unit-}j)\, Y_i(s)\, ds \tag{6}$$

All N^2 such means can be estimated together in a matrix $E \stackrel{\text{def}}{=} \{E_{ji}\}$. E^* is defined similarly. Table 1 shows a submatrix of E^* for our example system. The diagonal elements are the row maxima. This is to be expected, since they were derived by evaluating strings using the same models that were used to generate them. Note, however, that this argument does not carry over to the columns; some diagonal elements are *not* the column maxima (e.g. $D_{/\text{ax}/}(Y_{/\text{ax}/})$). This reflects the fact that some models tend to generate more "agreeable" strings than others.

A rough feel for similarity between some phonemes can be gleaned from this data. For example, columns /ae/ and /ay/ are similar (compare them to column /g/), as are rows /ae/ and /ay/. This corresponds to the similarity between these two vowels.

For a more rigorous measure of distance between models, consider how the off-diagonal means differ from the diagonal element. Let

$$\text{DIST}(j;i) \stackrel{\text{def}}{=} E_{jj} - E_{ji} \tag{7}$$

and similarly for DIST*. Table 2 shows a submatrix of DIST*, derived from E^* by a single matrix operation. The distance between phonemes based on this table seems to be in strong agreement with our intuition and domain knowledge.

This measure can be used for clustering of larger speech units. It is superior to phonemic clustering, which considers phonemes as atomic units. For example, bat and pat are much more

	/ae/	/ax/	/ay/	/w/	/ng/	/g/	/sh/	/dd/
/ae/	0.00	2.29	3.15	8.91	6.40	9.40	10.34	7.45
/ax/	5.94	0.00	11.04	7.17	5.84	7.62	10.92	6.02
/ay/	2.19	3.82	0.00	8.82	8.15	10.03	11.31	9.33
/w/	11.10	5.63	12.00	0.00	8.66	7.54	13.64	8.03
/ng/	7.95	4.48	10.17	7.86	0.00	6.06	10.37	4.61
/g/	13.52	8.77	14.91	8.28	9.54	0.00	10.29	2.16
/sh/	13.34	11.30	13.80	12.72	11.41	8.68	0.00	7.59
/dd/	15.21	12.23	16.40	13.21	11.09	8.41	13.10	0.00

Table 2: A submatrix of $\mathrm{DIST}^*_{ji} \stackrel{\mathrm{def}}{=} E^*_{jj} - E^*_{ji}$, a derived measure of distance between models.

similar acoustically than phonetically [1].

Recall that, for our HMM based system, we defined $Y_i(s) \stackrel{\mathrm{def}}{=} \frac{1}{|s|} \log P(s|M_i)$. For simplicity, let us write $P_i(s)$ for $P(s|M_i)$. Then

$$\begin{aligned} \mathrm{DIST}^*(i;j) &\stackrel{\mathrm{def}}{=} E^*_{jj} - E^*_{ji} \\ &= \int \frac{1}{|s|} P_j(s) \log P_j(s) ds - \int \frac{1}{|s|} P_j(s) \log P_i(s) ds \\ &= \int \frac{1}{|s|} P_j(s) \log \frac{P_j(s)}{P_i(s)} ds \end{aligned} \tag{8}$$

The last expression is similar to the "Asymmetric Divergence" — a well known measure of distance between two distributions[Kullback 59][2]. The difference is in the presence of the $\frac{1}{|s|}$ factor. Asymmetric Divergence was proposed as a measure of distance between HMMs by [Juang & Rabiner 85]. They derived it from information theoretic arguments. [D'orta et al. 87] used their measure, with a sampling technique similar to ours, to cluster phonemes. In our derivation, both the measure and the estimation method naturally "fall out" of the definition of the $D_j(Y_i)$'s. In addition, our definition is not limited to HMMs.

4. Main application: reducing search in large vocabularies

4.1. Changing the Classifier System

Given a acoustic string, in the traditional method of classification we ask the question:

> Which of the models $M_1, M_2, \ldots, M_N$ is the most likely to have produced s? (i.e., find the i that maximizes $Y_i(s)$.)

[1] We are grateful to Raj Reddy for this example.

[2] To make it symmetric, define $\mathrm{DIST}'(P_i, P_j) \stackrel{\mathrm{def}}{=} \mathrm{DIST}^*(P_i, P_j) + \mathrm{DIST}^*(P_j, P_i)$.

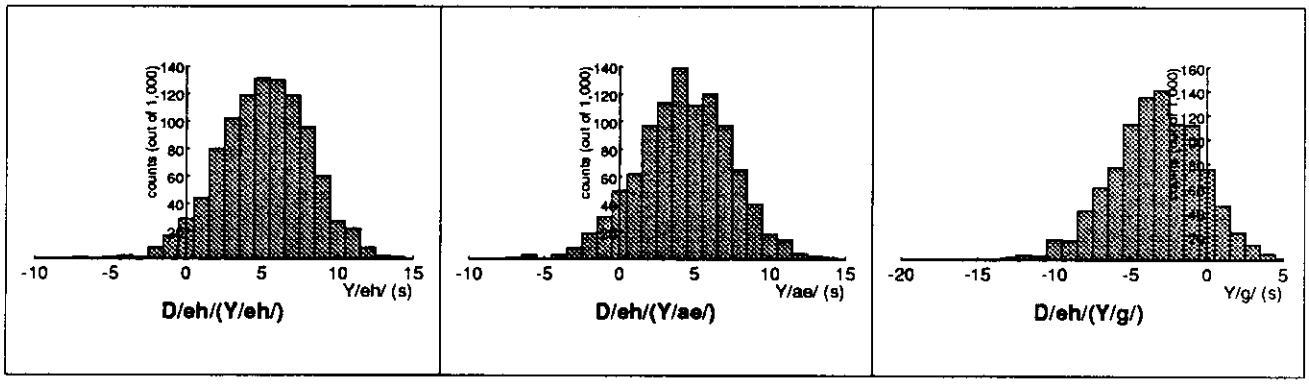

Figure 2: Histograms of some typical $D_j^*(Y_i)$'s, each based on a sample of 1000 strings generated from the HMM model M_j.

We now propose to change the question to:

> Which of the distributions $D_1(\mathbf{Y}), D_2(\mathbf{Y}), \ldots, D_N(\mathbf{Y})$ is the most likely to have produced s? (i.e., find the j that maximizes $D_j(\mathbf{Y}(s))$.)

The distributions $D_1(\mathbf{Y}), D_2(\mathbf{Y}), \ldots, D_N(\mathbf{Y})$ take the place of the models $M_1, M_2, \ldots, M_N$ as the Bayesian classifier. We do not necessarily expect the D_j's to perform as well as the M_i's. In fact, in our example system, Since $M_1, M_2, \ldots, M_N$ were optimized as Maximum Likelihood classifiers and not as discriminators, we expect a degradation of performance. However, this change of classifiers is a necessary first step towards reducing the search effort.

Since $D_j(\mathbf{Y}(s))$ is an N-dimensional distribution, an unrestricted non-parametric estimation is impractical for even a large sample. Some assumptions have to be made. There are many ways to proceed. Here we chose to assume that the individual $D_j(Y_i)$s are independent. This is clearly an incorrect assumption, as our data (and intuition) indicate. In making this assumption we are merely choosing to concentrate on the first-order statistics of the $D_j(Y_i)$ s, and to ignore for the time being the second- and higher-order relationships.

Thus we are looking for the j that maximizes $\prod_i D_j(Y_i)$. This still leaves us with the problem of estimating the N^2 distributions $D_j(Y_i)$. What do these distributions look like? Figure 2 shows histograms of selected $D_j^*(Y_i)$'s, each based on a sample of 1000 strings, which were generated from the appropriate model.

The distributions are well characterized by a Normal (Gaussian) curve. This is true for all the distributions we checked. In retrospect, it is not difficult to see why this happens. Since the strings were generated from Hidden Markov Models, each frame in each codebook was drawn independently. Therefore, $\log P(s|M_i)$ is a sum of many independent events, hence the Gaussian curve.

The Normal shape of the distributions is welcome news, because they can be characterized fairly well with only two parameters each: mean and standard deviation. These can be estimated

	/ae/	/ax/	/ay/	/w/	/ng/	/g/	/sh/	/dd/
/ae/	7.7 ± 3.7	5.4 ± 3.2	4.5 ± 6.6	-1.2 ± 3.6	1.3 ± 4.0	-1.7 ± 3.3	-2.7 ± 4.2	0.2 ± 3.3
/ax/	-5.3 ± 7.8	0.7 ± 3.8	-10.4 ± 8.3	-6.5 ± 4.4	-5.2 ± 4.7	-6.9 ± 4.2	-10.2 ± 4.5	-5.3 ± 4.2
/ay/	6.8 ± 3.1	5.1 ± 2.5	9.0 ± 3.0	0.1 ± 2.8	0.8 ± 3.1	-1.1 ± 2.7	-2.4 ± 3.4	-0.4 ± 2.7
/w/	-9.6 ± 4.2	-4.1 ± 2.9	-10.4 ± 5.2	1.6 ± 2.7	-7.1 ± 3.5	-6.0 ± 2.8	-12.1 ± 3.3	-6.5 ± 2.9
/ng/	-3.7 ± 4.9	-0.2 ± 3.0	-5.9 ± 5.2	-3.6 ± 3.3	4.2 ± 2.8	-1.8 ± 3.0	-6.1 ± 3.3	-0.4 ± 2.7
/g/	14.0 ± 3.9	-9.2 ± 3.4	-15.3 ± 3.8	-8.7 ± 3.4	-10.0 ± 4.2	-0.4 ± 2.8	-10.7 ± 3.6	-2.6 ± 3.1
/sh/	-5.5 ± 4.3	-3.5 ± 3.4	-6.0 ± 4.4	-4.9 ± 3.6	-3.6 ± 3.7	-0.9 ± 3.0	7.8 ± 3.4	0.2 ± 3.0
/dd/	15.4 ± 4.8	-12.4 ± 6.5	-16.6 ± 4.1	-13.4 ± 5.8	-11.3 ± 7.3	-8.6 ± 9.2	-13.3 ± 5.9	-0.2 ± 4.8

Table 3: A submatrix of $E^* \pm \sigma^*$ (the means and standard deviations of the $D_j^*(Y_i)$'s).

	Top 1	Top 2	Top 3	Top 4	Top 7	Top 10	Top 15	Top 20
Ranking by P_1	24%	37%	46%	54%	70%	79%	88%	94%

Table 4: Performance of the $D_1^*, D_2^*, \ldots, D_N^*$ classifiers, using first-order statistics only. For example, in 79% of the strings tested, the generating model was ranked among the top 10 contenders using equation 9.

accurately and reliably from a modest sample. Note that, in other models, if the $D_j(Y_i)$'s are not gaussians, accurate characterization may be more difficult. However, the mean and standard deviation can still be used to derive statistical bounds. The resulted inference is expected to be weaker, though.

Assuming $D_j(Y_i) \sim \mathcal{N}(E_{ji}, \sigma_{ji})$, classification can now be done by finding the j that minimizes:

$$-\log P_1(\mathbf{Y}|D_j) = \sum_{i=1}^{N} \left[\log \sigma_{ji} + \frac{(Y_i - E_{ji})^2}{2\sigma_{ji}^2} \right] \tag{9}$$

Where the subscript "1" denotes the use of first-order statistics only.

Table 3 shows the estimated $E^* \pm \sigma^*$ values for a submatrix of our example system. The full matrix can be kept in main memory, making the computation of equation 9 straightforward and inexpensive (assuming $\mathbf{Y}$ is known).

4.2. Performance of the New Classifier

How good is our new classifier? Since we made many simplifying assumptions, and since $M_1, M_2, \ldots, M_N$ were derived as Maximum Likelihood models and not as discriminators, we do not expect the performance of $D_1, D_2, \ldots, D_N$ to be nearly as good as that of the original classifier. However, it is still instructive to gauge it. Table 4 lists one possible measure of performance.. The 24% figure under "Top 1" means that in 24% of the strings tested, the generating models was correctly given the highest P_1 value by our new classifier. In 37% of the cases, it was ranked among the top 2 contenders, and so on (the percentages are cumulative).

	Top 1	Top 2	Top 3	Top 4	Top 7	Top 10	Top 15	Top 20
P_1 Ranking	24%	37%	46%	54%	70%	79%	88%	94%
Normalized P_1 Ranking	66%	80%	87%	91%	96%	98%	99.5%	99.8%

Table 5: Performance of the $D_1^*, D_2^*, \ldots, D_N^*$ classifiers, using first-order statistics only, for unnormalized and normalized strings.

Although these results are interesting, they are clearly not sufficient for decision making. How can they be improved? The accuracy and reliability of any Bayesian classifier depends crucially on how well separated the class distributions are. In our context, this translates into the ratio of between-string variability to between-model variability.

A close look at our data reveals very significant between-string variability. Some strings receive good scores from all the models, while others receive bad scores. There is very significant correlation between the various scores given to the same string. In fact, we found the pairwise correlation coefficients to lie in the range 0.93–0.99, regardless of the distribution from which the strings came, or the pair of models used for evaluation. This "global correlation" among the Y_i's of the same string means that some strings are "better acoustic segments" than others. This may be because some speech frames are further away from the codeword centers, resulting in weak matches with all models alike.

To get rid of most of this "global correlation", we *normalize* the scores by subtracting, from each $Y_i(s)$, the average $\overline{Y}(s)$ of $Y_1(s), Y_2(s), \ldots, Y_N(s)$. By subtracting the average, we eliminate that portion of the correlation that is due to the intrinsic "goodness" of the string. Of course, the $E \pm \sigma$ table needs to be modified similarly. Our new results are listed in table 5. The improvement is indeed very significant. Our new classifier can now be used in the following manner: if we desire, say, a 96% confidence in the classification decision, we restrict our attention to the models that were ranked 1–7 by our new classifier, and choose the one with the highest $Y_i(s)$ among them.

Of course, so far we did not reduce the computational requirements, because in order to compute $\log P_1(\mathbf{Y}|M_j)$ according to equation 9, we must first know $\mathbf{Y}(s) \stackrel{\text{def}}{=} Y_1(s), Y_2(s), \ldots, Y_N(s)$. In the next section we discuss how to overcome this problem.

4.3. Estimating the Scores

In order to avoid computing all N scores $Y_1(s), Y_2(s), \ldots, Y_N(s)$, we *estimate* $\log P_1(\mathbf{Y}(s)|M_j)$ using only a subset of the Y_i values. We view this subset as a sample of the entire model population. Since this is just an estimate, some degradation of performance is likely. We expect performance to get worse as the size of the sample decreases. Table 6 shows performance of the estimated classifier, with different sample sizes. All samples were drawn randomly and independently for every test string. As we expected, there is a clear trade-off between sample size and performance.

There is one more problem. We mentioned that the data is normalized by subtracting $\mathbf{Y}(s)$'s average. But in order to know that average, we must again compute all of $Y_1, Y_2, \ldots, Y_N$. We

	Top 1	Top 2	Top 3	Top 4	Top 7	Top 10	Top 15	Top 20
Normalized P_1 Ranking	66%	80%	87%	91%	96%	98%	99.5%	99.8%
Estimating P_1								
sample size = 24	62%	77%	85%	89%	95%	98%	99.5%	99.9%
sample size = 16	59%	75%	83%	88%	94%	97%	99.3%	99.8%
sample size = 10	53%	70%	79%	85%	93%	97%	99.0%	99.7%
sample size = 8	50%	67%	77%	83%	92%	96%	99.0%	99.7%
sample size = 6	44%	62%	72%	79%	90%	95%	98.5%	99.6%
sample size = 4	37%	54%	65%	72%	86%	92%	97%	99.0%

Table 6: Performance of the *estimated* D_i^* classifiers, using first-order statistics only. Exact (non-estimated) ranking is included for comparison.

	Top 1	Top 2	Top 3	Top 4	Top 7	Top 10	Top 15	Top 20
Normalized P_1 Ranking	66%	80%	87%	91%	96%	98%	99.5%	99.8%
Estimating P_1 & $\overline{Y}$								
sample size = 16	54%	71%	79%	84%	92%	96%	98.6%	99.5%
sample size = 10	43%	59%	69%	75%	86%	91%	96%	98.0%
sample size = 8	37%	53%	62%	69%	81%	88%	94%	97%

Table 7: Performance of the *estimated* D_i^* classifiers, where the normalization stage too was based on an *estimated* average. Exact (non-estimated) ranking is included for comparison.

solve this problem by again estimating the average $\overline{Y}$ from a sample. We use the same independent random sample to estimate both $\overline{Y}$ and the $P_1(\mathbf{Y})$'s. Table 7 summarizes our results.

As expected, there is further degradation, but this time there are no more hurdles. We now have a probabilistic algorithm for finding the Maximum Likelihood model, which will give the correct answer with any desired confidence level. The algorithm saves us some work over computing all N score values. The savings are not dramatic, but there are good reasons why they should increase considerably when our classifier is applied to real-world, large vocabulary systems:

1. For a given level of accuracy and confidence, the necessary sample size does not depend on the size of the population — it is only a function of the variance of the data. The latter can be expected to remain fixed as the vocabulary increases, because the dynamic range of the estimated values remains the same. A sample of size 20 is large relative to a vocabulary of size 48, but the same sample size represents significant savings for a vocabulary of 1,000 or more items.

2. When the models are longer (words as opposed to phonemes), they are more distinct, and therefore the between-model variance is greater, resulting in better classification rate.

On the other hand, it is yet to be seen whether our approach will work on real speech and large vocabularies. Some possible problems are:

P_1 Ranking:	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
String 1:	01	03	02	04	05	11	10	14	12	07	06	08	09	19	18	15	16	31	24	38
String 2:	03	01	04	02	09	05	31	08	29	21	07	18	36	10	06	30	13	15	27	11
String 3:	01	02	03	16	04	07	08	12	09	05	32	13	10	06	15	19	11	22	23	31
String 4:	01	03	04	02	06	11	10	09	05	13	17	16	07	08	18	14	22	21	19	12
String 5:	02	03	01	10	16	04	09	06	12	11	05	14	21	18	13	07	19	27	20	23
String 6:	01	07	13	09	02	08	10	06	11	17	16	03	04	12	18	05	31	25	24	30
String 7:	03	10	01	11	12	08	02	05	21	06	13	15	04	14	20	07	09	16	18	40
String 8:	01	19	46	32	11	16	48	29	06	30	08	05	20	38	23	25	15	14	03	33
String 9:	01	19	20	28	31	12	16	33	25	24	26	22	11	04	06	29	21	23	18	27
String 10:	01	35	19	17	11	08	05	16	20	47	14	21	28	23	13	06	02	38	15	09

Table 8: The "true" (ML) ranking of the 20 models which were ranked highest by the new classifier, for ten randomly selected acoustic strings.

1. Real speech is different from the synthetic frames we generated for the experiment above. The distributions D_i are different from the D_i^*'s, and may be more difficult to characterize or to separate.

2. In a large vocabulary, a given entry is on the average confusable with more other entries than in our small test system.

We plan to test all of these assumption when we implement our approach on the $1,000$ word Resource Management database.

4.4. Other Measures of Performance

The performance measure discussed above is but one way of measuring the usefulness of our method. It is appropriate, for example, in isolated word recognition systems, where the classifier is used to help decode a given, isolated word. In other contexts, other measures of performance may be more appropriate. In continuous speech systems, for example, a lattice of word hypotheses is often generated by the recognizer, and used by higher level processes. Many possible measures may be used to gauge the performance of a classifier with regard to this goal. The correct measure depends on the way the lattice will be used. Here we chose not to commit to a specific numerical measure. Instead, we merely display a few typical results. Each row in table 8 displays the "true" (i.e. Maximum Likelihood) ranking of 20 of the 48 models. These are the models ranked highest by our P_1 classifier, sorted in descending order of P_1 value. (For example, in the first string, the model which was ranked 8^{th} by the new classifier turned out to have the 14^{th} highest Y score.) For most inputs, all of the top 10 ML models were included by the new classifier in its top 20 list.

4.5. Potential Improvements

The results discussed above are preliminary. The following can be used to try to achieve further improvement:

Judicious choice of the sample: In the experiments described in the previous subsection, we chose a new sample randomly for every string. Undoubtedly this is not optimal. We can use statistical analysis (e.g. multiple regression) to choose the subset that best predicts P_1 and $\overline{Y}$. This has the added advantage of allowing us to keep in memory only those columns of the $[E, \sigma]$ table that correspond to that subset. For large vocabularies, this represents a significant saving in memory requirements.

Using higher-order statistics: So far we discussed and exploited only the first-order behavior of the distributions $D_j(\mathbf{Y})$. Higher-order statistics can also be employed. Much more information can be gleaned from even the second-order behavior alone. If two models are similar, than a string scoring well (badly) on one is likely to score well (badly) on the other. For two very different models, a good score on one implies a bad score on the other. These deductions are based on a generalized form of the Triangle Inequality, although they do not require that the distance between the models be a metric. An elimination algorithm similar to that reported in [Vidal et al. 88] can then be used to implement Fast Search.

Better normalization: The normalization we used in order to reduce the global correlation is an ad-hoc subtraction of the string's average score. Better methods may be possible, leading to lower within-string variance, and hence to better performance[3].

Better modeling of the $D_j(\mathbf{Y})$'s: This may be particularly useful when the distributions are estimated from real speech samples $(D_1, D_2, \ldots, D_N)$ and not from strings generated by the models $(D_1^*, D_2^*, \ldots, D_N^*)$. We expect the former to match the Gaussian curve less well than the latter do.

Other statistics of s: We can view $\mathbf{Y}(s)$ as a set of statistics of the acoustic string s, which reduce its dimensionality to a reasonable level. There is no reason why $\mathbf{Y}$ should not include other statistics of s as well. One plausible candidate is the string's length, namely the number of speech frames it has. Other statistics can be suggested.

5. Speculations on other uses

In section 2 we developed a general formalism for exploiting dependencies among competing models. The most obvious application of this formalism, namely reducing search in large vocabularies, was discussed extensively above. In this section we speculate about other possible uses for this framework. We are motivated here by the observation that $\mathbf{Y}(s)$ is a source of information about

[3] We did try, unsuccessfully, to use the *length* of the strings (number of frames) to normalize the scores. We found very little correlation between the two.

the incoming string that is made available to us at no extra cost during a traditional recognition phase. This information has always been around, but to the best of our knowledge has never been used.

Improving on the maximum likelihood classifier: The "correct" measure relative to which all classifiers should be judged is $P(s|s \in \text{speech-unit-}j)$. The traditional maximum-likelihood classification can be viewed as an approximation of this measure. The same goes for our method, which attempts to model the above measure using the $D_j(\mathbf{Y})$'s. Neither model is perfect, and it is conceivable that under some circumstances, our classifier may be superior. This may happen, for instance, if we modify it to weigh the Y_i's differently. The traditional ML method can then be seen as a special case where Y_j has a weight of 1, and all the others a weight of 0. It is possible that other weight vectors will perform better.

Helping detect a low-confidence classification: Currently, SPHINX's misclassifications are difficult to detect automatically. The distance between the best and second-best match scores is apparently not a good indicator of successful classification. It is possible that the additional information provided by the Y_i's will improve our ability to attach a level of confidence to the recognizer's output. One possible way of doing that is through a "confusion matrix" derived from the $D_j(\mathbf{Y})$'s.

Acknowledgments

We received helpful comments and encouragement from Raj Reddy, Kai-Fu Lee and Fil Alleva. Much appreciated feedback was also provided by Rich Stern and Sunil Kumar. Dan Julin generously shared with us his expertise in various software areas. We are grateful to all of them. This work was done in partial fulfillment of the first author's AQ requirements.

References

[Aubert 89] Aubert, X. "Fast Look-Ahead Pruning Strategies in Continuous Speech Recognition". *Proc. ICASSP 89*, pp. 659–662, Glasgow, Scotland, May 1989.

[Bahl et al. 88a] Bahl, L. R., Brown, P. F., de Souza, P. V., and Mercer, R. L., "A New Algorithm for the Estimation of Hidden Markov Parameters". *ICASSP 88*, pp. 493–496, April 1988.

[Bahl et al. 88b] Bahl, L., Bakis, R., de Souze, P., and Mercer, R. "Obtaining Candidate Words by Polling in a Large Vocabulary Speech Recognition System". *Proc. ICASSP 88*, pp. 489–492, New-York, NY, April 1988.

[Bahl et al. 89a] Bahl, L., Gopalakrishnan, P.S., Kanevsky, D., and Nahamoo, D. "Matrix Fast Match: A Fast Method for Identifying a Short List of Candidate Words for Decoding". *Proc. ICASSP 89*, pp. 345–347, Glasgow, Scotland, May 1989.

[Bahl et al. 89b] Bahl, L., de Souze, P., Gopalakrishnan, P.S., Kanevsky, D., and Nahamoo, D. "Constructing Groups of Acoustically Confusable Words Candidate Words for Decoding". Proc. ICASSP 89, pp. 345–347, Glasgow, Scotland, May 1989.

[Casacuberta et al. 87] Casacuberta, F., Vidal, E., and Rulot, H. "On the Metric Properties of Dynamic Time Warping". *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-35, pp. 1631–1633, November 1987.

[D'orta et al. 87] D'orta, P., Ferretti, M., and Scarci, S., "Phoneme Classification for Real Time Speech Recognition of Italian". *Proc. ICASSP 87*, pp. 81–84, Dallas, TX, 1987.

[Huang et al. 90] Huang, X., Lee, K., and Hon, H. "On Semi-Continuous Hidden Markov Models". *ICASSP 90*, pp. 689–692, 1990.

[Juang & Rabiner 85] Jaung, B. H., and Rabiner, L. R., (1985). "A Probabilistic Distance Measure for Hidden Markov Models". *AT&T Technical Journal* 64(2).

[Kaneko & Dixon 83] Kaneko, T., and Dixon, N. "A Hierarchical Decision Approach to Large-Vocabulary Discrete Utterance Recognition". *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-31, pp. 1061–1066, October 1983.

[Kullback 59] Kullback, S. *Information Theory and Statistics.* New York, Wiley, 1959.

[Lee 88] Lee, K.F. *Large-Vocabulary Speaker-Independent Continuous Speech Recognition: The SPHINX System.* PhD thesis, Computer Science Department, Carnegie Mellon University, April 1988.

[Sakoe & Chiba 78] Sakoe, H., and Chiba, S., "Dynamic programming algorithm optimization for spoken word recognition." *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-26(1):43–49, February 1978.

[Vidal et al. 88] Vidal, E., Rulot, H., Casacuberta, F., and Benedí, J. M. "On the Use of Metric-Space Search Algorithm (AESA) for Fast DTW-Based Recognition of Isolated Words". *IEEE Trans. Acoust., Speech, Signal Processing,* vol. ASSP-36, pp. 651–656, May 1988.