EXPLOITING CORRELATIONS AMONG COMPETING MODELS WITH APPLICATION TO LARGE VOCABULARY SPEECH RECOGNITION

Ronald Rosenfeld, Xuedong Huang and Merrick Furst

School of Computer Science
Carnegie Mellon University
Pittsburgh, Pennsylvania 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.

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 [1] and some metric distance defined over frames. In HMM-based systems, a match is typically defined as the class-conditional log-probability (log $P(s|M_i)$) of the speech model and an acoustic instance.

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

1. Computing the match for all the models is computationally expensive. For large vocabularies, 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. Attempts have been made recently to overcome this problem by using fast preliminary search ([3, 4, 5, 6]). Here, we take a different approach to the problem. We propose to make use of all computed scores, by means of statistical inference. In order to...
3 DERIVING A MEASURE OF DISTANCE BETWEEN MODELS

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

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

$$E_{jj} \overset{\text{def}}{=} E[D_j(Y_i)] = \int P(s|\text{speech-unit};j) Y_i(s) \, ds$$  \hspace{1cm} (6)

All $N^2$ such means can be estimated together in a matrix $E \overset{\text{def}}{=} \{E_{ij}\}$. $E^*$ 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. However, note that this argument does not carry over to the columns; some diagonal elements are not the column maxima (e.g. $D_{/ax/}(Y_{/ay/})$). 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

$$\text{DIST}^*(i,j) \overset{\text{def}}{=} E_{jj} - E_{ji}$$  \hspace{1cm} (7)

and similarly for $\text{DIST}^*$.

This measure can be used to cluster larger speech units. It is superior to phonemic clustering, which considers phonemes as atomic units. For example, $\text{bat}$ and $\text{pat}$ are more similar acoustically than phonetically.

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

$$\text{DIST}^*(i,j) \overset{\text{def}}{=} E_{jj}^* - E_{ji}^*$$  \hspace{1cm} (8)

$$= \int \frac{1}{|s|} P_i(s) \log P_j(s) ds - \int \frac{1}{|s|} P_i(s) \log P_i(s) ds$$

$$= \int \frac{1}{|s|} P_i(s) \log \frac{P_j(s)}{P_i(s)} ds$$

We are grateful to Raj Reddy for this example.

The last expression is similar to the "Asymmetric Divergence" — a well known measure of distance between two distributions[9].

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 and Rabiner [10]. They derived it from information theoretic arguments. D'orta et al.[11] 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. More importantly, our definition is not limited to HMMs.

4 REDUCING SEARCH IN LARGE VOCABULARIES

4.1 Changing the Classifier

As a first step towards reducing the search effort, we replace the original Bayesian classifier $M_1, M_2, \ldots, M_N$ with a new one, $D_1(Y), D_2(Y), \ldots, D_N(Y)$.

Since $D_j(Y(s))$ is an $N$-dimensional distribution, an unrestricted non-parametric estimation is impractical for even a large sample. We proceed by assuming that the individual $D_j(Y_i)$s are independent. This is clearly incorrect, 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. More importantly, our definition is not limited to HMMs.

Table 1: A submatrix of $E^{\text{def}} = E^*_j$ (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.

<table>
<thead>
<tr>
<th>$/ae/$</th>
<th>$/ax/$</th>
<th>$/ay/$</th>
<th>$/w/$</th>
<th>$/ng/$</th>
<th>$/g/$</th>
<th>$/sh/$</th>
<th>$/dd/$</th>
</tr>
</thead>
<tbody>
<tr>
<td>7.68</td>
<td>5.39</td>
<td>4.53</td>
<td>-1.23</td>
<td>1.28</td>
<td>-1.72</td>
<td>-2.66</td>
<td>0.23</td>
</tr>
<tr>
<td>-5.25</td>
<td>0.68</td>
<td>-10.35</td>
<td>-6.49</td>
<td>-5.15</td>
<td>-6.94</td>
<td>-10.24</td>
<td>-5.33</td>
</tr>
<tr>
<td>6.76</td>
<td>5.13</td>
<td>8.95</td>
<td>0.13</td>
<td>0.80</td>
<td>-1.08</td>
<td>-2.36</td>
<td>-0.38</td>
</tr>
<tr>
<td>-9.55</td>
<td>-4.08</td>
<td>-10.45</td>
<td>1.55</td>
<td>-7.11</td>
<td>-5.99</td>
<td>-12.09</td>
<td>-6.48</td>
</tr>
<tr>
<td>-3.70</td>
<td>-0.23</td>
<td>-5.93</td>
<td>-3.62</td>
<td>4.25</td>
<td>-1.81</td>
<td>-6.13</td>
<td>-0.37</td>
</tr>
<tr>
<td>-13.96</td>
<td>-9.20</td>
<td>-15.34</td>
<td>-8.71</td>
<td>-9.98</td>
<td>-0.44</td>
<td>-10.73</td>
<td>-2.60</td>
</tr>
<tr>
<td>-5.53</td>
<td>-3.50</td>
<td>-6.00</td>
<td>-4.91</td>
<td>-3.60</td>
<td>-0.87</td>
<td>7.80</td>
<td>0.21</td>
</tr>
</tbody>
</table>
used to derive statistical bounds. The resulted inference is expected
to be weaker, though.

Assuming \( D_j(Y_i) \sim N(\mu_j, \sigma_j) \), classification can now be done
by finding the \( j \) that minimizes:

\[
-\log P_j(Y_i|D_j) = \sum_{i=1}^{N} \left[ \log \sigma_i + \frac{(Y_i - \mu_j)^2}{2\sigma_i^2} \right]
\]

(9)

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

4.2 Performance of the New Classifier

How good is the new classifier? If the \( M_j \)'s are ML classifiers, some
performance degradation is expected. Table 2 lists one possible
measure of performance. In the first row, in 24% of the strings
tested, the generating models was correctly given the highest \( P_j \)
value by the new classifier. In 37% of the cases, it was ranked among
the top 2 contenders, and so on (the percentages are cumulative).

How can these results be improved? The performance 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
confusable". Less confusable. Their between-model variance is therefore
greater, resulting in better classification rate.

To get rid of most of this global correlation, we normalize
the scores by subtracting, from each \( Y_i(s) \), the average \( Y(s) \) of
\( Y_1(s), Y_2(s), \ldots, Y_N(s) \). The normalized results are listed in the
second row of table 2. The improvement is indeed very significant.

4.3 Estimating the Scores

To avoid computing all \( N \) scores \( Y_1(s), Y_2(s), \ldots, Y_N(s) \), we estimate
log \( P_j(Y(s)|M_j) \) and \( Y(s) \) using only a subset of the \( Y_i \) values.

Figure 1: Histograms of some typical \( D_j(Y_i) \)'s, each based on a sample of 1000 strings generated from the HMM model \( M_j \).

Performance degradation will depend on the sample size. The last
part of table 2 shows the performance of the estimated normal-
ized classifier, for different sample sizes. All samples were drawn
randomly and independently for every test string.

4.4 Using the New Classifier

To quickly obtain short lists of candidate words, we use the fol-
lowing algorithm: compute \( Y(s) \) for a small random subset of the
models, and output the models ranked highest by this estimated
classifier.

To reduce the effort in searching for the top ML model, we use the
following probabilistic algorithm: if we desire, say, a 96%
confidence in the classification, we compute \( Y(s) \) for 10 randomly
selected models, restrict our attention to the models that were ranked
1-15 by the estimated classifier, and choose the one with the highest
\( Y \) among them. We only need to compute a total of 22 match scores
on average, and no more than 25.

The two algorithms above save us some work over computing all
\( N = 48 \) score values. For our small test system the savings are not
dramatic, but they should increase considerably when the 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, which is fixed.
A sample of size 20 is large relative to our test vocabulary of
48 models, but represents significant savings for a vocabulary
of, say, 1,000 items.

2. longer models (i.e. words as opposed to phonemes) are
less confusable. Their between-model variance is therefore
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:

1. Real speech is different from the synthetic frames we gen-
erated for the experiment above. The distributions \( D_i \) are
different from the \( D_i^* \)'s, and may be more difficult to charac-
terize or to separate.

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

I-7
We plan to test these assertions when we implement this approach on a large vocabulary system, such as RM or WSJ.

4.5 Potential Improvements

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

Judicious choice of the sample: In the experiments described above, 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 $D(Y)$. 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 [12] 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.

Better modeling of the $D(Y)$’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 $Y(s)$ as a set of statistics of the acoustic string $s$, which reduces its dimensionality to a reasonable level. There is no reason why $Y$ 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.

---

### Table 2: Performance of the $D'_s$ classifier, based on first-order statistics only. $N = 48$.

| $P_1$ Ranking | Top 1 | Top 2 | Top 3 | Top 4 | Top 5 | Top 6 | Top 7 | Top 8 | Top 9 | Top 10 | Top 11 | Top 12 | Top 13 | Top 14 | Top 15 | Top 20 |
|---------------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|
|               | 24%   | 37%   | 46%   | 54%   | 70%   | 79%   | 88%   | 94%   |       |       |       |       |       |       |       |       |
| Normalized $P_1$ Ranking | 66%   | 80%   | 87%   | 91%   | 96%   | 98%   | 99%   | 99%   |       |       |       |       |       |       |       |       |
| Estimated $P_1$ Ranking |       |       |       |       |       |       |       |       |       |       |       |       |       |       |       |       |
| sample size = 16 | 54%   | 71%   | 79%   | 84%   | 92%   | 96%   | 98.6% | 99.5% |       |       |       |       |       |       |       |       |
| sample size = 10 | 43%   | 59%   | 69%   | 75%   | 86%   | 91%   | 96%   | 98%   |       |       |       |       |       |       |       |       |
| sample size = 8  | 37%   | 53%   | 62%   | 69%   | 81%   | 88%   | 94%   | 97%   |       |       |       |       |       |       |       |       |

---

### ACKNOWLEDGMENTS

We are grateful to Raj Reddy, Kai-Fu Lee, Fil Alleva and Dan Julin for helpful comments and encouragement. This research was sponsored by the Defense Advanced Research Projects Agency (DOD), ARPA Order No. 7239, under contract number N00039-91-C-0158.

### References


