Two-Level Phoneme Recognition Based on Successive Use of Monophone and Diphone Models

Panu Somervuo
Neural Networks Research Centre
Helsinki University of Technology
P.O.Box 5400, FIN-02015 HUT
Finland
panu.somervuo@hut.fi

ABSTRACT

Two-level phoneme recognition method is proposed based on successive use of monophone and diphone models. In the first level of the recognition, computationally lighter (in terms of the number of the models) monophone models are used for selecting a subset of diphone models. For each input utterance, those diphone models are set active whose left or right contexts are present in the recognized monophone sequence. The chosen diphone models are then evaluated in the next level of the recognition. This substantially decreases the computational load compared to the case where all diphone models must be examined for each input utterance. In the Finnish speaker-independent phoneme recognition task on average half of the diphone models could be eliminated in the second level of the recognition per word utterance while still achieving the same recognition accuracy as when using all the models. Clustered monophone and diphone models were also experimented as the models in the first-level recognizer. This did not, however, bring any further improvement to the results obtained by using unclustered monophone and diphone models.

1 Introduction

The approach to the current state-of-the-art speech recognition can be described as a task for finding the best path in a large graph where the nodes represent atomic elements of speech [6]. It is important to improve both the models of the nodes in the graph, i.e., the current acoustic models, and the restrictions to the best path in the graph, i.e., the language models. However, in addition to these fundamental modeling tasks, it is also important to develop speedup methods for the recognition systems in order to be able to construct practical applications.

Expanding N-grams can be applied to both acoustic and language models. The common goal is to utilize the better accuracy of the more specific models while avoiding unnecessary computation. Short-cuts are needed in order to keep the computational load feasible. One approach is to use hierarchical recognition system for limiting the search space. Recognition process begins using coarse models and then proceeds using finer and more detailed models in the next level. In this work monophone and diphone models have been used, see Fig. 1.

A common method in continuous speech recognition is to use Viterbi-beam search. Only recognition hypotheses above a certain likelihood threshold are evaluated further. This reduces the amount of computation. An obvious problem, however, is how to choose the proper threshold for cutting the unlikely hypotheses. In the present work where the goal is phoneme recognition no such threshold needs to be explicitly determined. The result of the monophone recognizer is used for automatically selecting a subset of diphone models for being evaluated.

The aim of the phoneme recognition is to convert a feature vector sequence into a phoneme symbol sequence. In the Finnish language where the orthography of a word is almost the same as its phonemic representation, a continuous phoneme recognizer will provide means for a vocabulary-free dictation system.

2 Speech data

Speech database consisted of word utterances collected from 59 native Finnish speakers. Each speaker had read a newspaper article in an isolated-word mode. The database was randomly divided into three sets: training set consisting of 30 speakers, development set consisting of 17 speakers, and final test set consisting of the remaining 12 speakers. These sets contained 7298, 4290, 3000 and words, and 61907, 33768, and 23863 phonemes, respectively. The phoneme set consisted of 24 symbols including all existing phonemes in the Finnish language. No distinctions were made between long and short versions of the same phoneme.

All data were recorded in a quiet office environment using a 16 kHz sampling rate. 24-dimensional feature vectors were computed from overlapping 16-ms time windows. High frequencies were emphasized using a first-order high-pass filter (zero at z=0.95) and the resulting speech samples were weighted using a Hamming window. Feature vectors consisted of 12 mel-

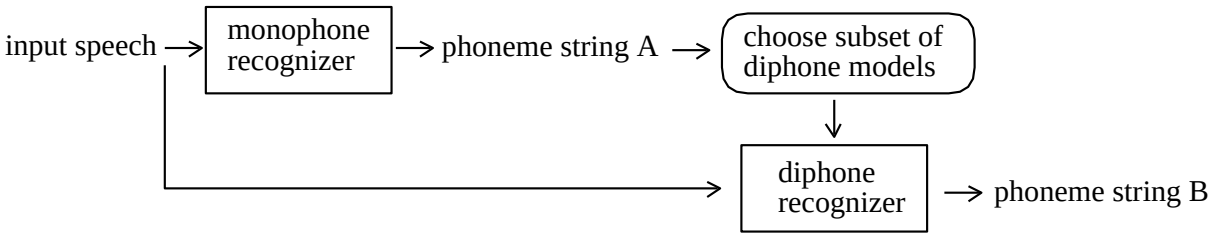

Figure 1: *Two-level phoneme recognition. Based on the result of the monophone recognition of an input utterance (phoneme string A), a subset of diphone models is selected to be active. The final recognition result based on diphone models is phoneme string B.*

frequency based cepstral coefficients and their first-order time derivatives. The time derivatives were computed using two previous and two ensuing speech frames.

3 Phoneme model training

The recognizer used in this work was based on hidden Markov models (HMMs) [7] with Gaussian mixture model densities. The mixture models were initialized using the Self-Organizing Map (SOM) and fine tuned with Learning Vector Quantization (LVQ) [2, 3].

The SOM can be thought as an elastic grid which is fitted to the input data. Each node is associated with a model of the input space (usually a feature vector). Through an unsupervised learning process the models become specially tuned and organized according to the input patterns smoothly approximating the distribution of the input data. The batch training of the SOM can be expressed as [3]

$$\mathbf{m}_i = \frac{\sum_j h_{c(\mathbf{x}_j),i}(t)\mathbf{x}_j}{\sum_j h_{c(\mathbf{x}_j),i}(t)}, \qquad (1)$$

where $\mathbf{m}_i$ is the model vector associated with the map unit i, $h_{c(\mathbf{x}_j),i}(t)$ is the neighborhood function, $c(x_j)$ is the index of the best-matching unit for data vector $\mathbf{x}_j$ (the model vector with the smallest Euclidean distance to the input vector), and t is the training time index. The neighborhood function controls the learning rates of the model vectors on the map. A large neighborhood is used in the beginning of the training which results in moving all model vectors near to the center of the input data even if the model vectors had been randomly initialized. As the training proceeds, the neighborhood function is then gradually set smaller so that the model vectors will better adapt to the finer details of the input data set.

In its basic formulation, the SOM algorithm organizes a vector quantization (VQ) codebook according to the similarity of static, separate feature vectors taking no temporal dependencies between the feature vectors into account. The benefit of the HMM is then utilizing the time-dependency and order of acoustic phenomena in the recognition. Temporal speech patterns are compactly stored in a state network. Combinations of the SOMs and HMMs have been used earlier e.g. in [5]. In short, SOM gives a good initialization for the mean vectors of the Gaussian mixture model.

Here spherical Gaussians were attached to each SOM node with a common kernel width. This corresponds to the reduced kernel-density estimator [1] with a single smoothing parameter among the kernels. The value of the smoothing parameter was set so that the number of the insertion and deletion errors using development data were in balance. The use of a single common smoothing parameter is advantageous, since it allows an easy way to use LVQ training when later fine tuning the mixture densities for better phoneme discrimination. Another aspect is that robust estimates for individual variances of Gaussians are difficult to obtain. The model used in the present work can be thought as a smoothed discrete probability density. The obvious advantage compared to the discrete pdf is the utilization of the quantization error between the observation vector and the VQ-codebook vector and the benefit compared against the full parametric continuous pdf model is the reduced number of parameters.

Phonemewise SOMs were initialized and trained using presegmented training data. It is possible to train a large SOM and then allocate kernels to phonemes according to majority voting. In this work, however, an equal amount of kernels was first allocated to each phoneme model. After training it is then possible to prune away all weakly used kernels.

Segmental K-means (SKM) algorithm [7] was used for re-training the models. In the K-means algorithm each data vector is used for updating only its closest model vector. Compared to the SOM algorithm, there is no neighborhood function (or it is the Kronecker delta). The updating of the model vectors is performed as

$$\mathbf{m}_i = \frac{\sum_j \delta(c(\mathbf{x}_j) - i)\mathbf{x}_j}{\sum_j \delta(c(\mathbf{x}_j) - i)}, \qquad (2)$$

where $\delta(c(\mathbf{x}_j)-i)$ is 1 if $\mathbf{m}_i$ is the closest model vector to $\mathbf{x}_j$ and 0 otherwise. Closest model vector to each data vector is sought in the codebook of that state which is assigned by the forced Viterbi-alignment corresponding the correct state sequence of the training data.

Finally, the models were fine tuned using the Learning Vector Quantization (LVQ) [2, 3] algorithm embedded in the Viterbi search [5]. In this error-corrective training process, two state sequences are sought for each training utterance. One sequence is the result of the unforced Viterbi search giving the most likely state sequence without any constraints of the correct recognition result. Another state sequence is obtained by the forced Viterbi search giving the best alignment of the state sequence corresponding to the correct phoneme symbol sequence of the utterance. LVQ is then applied frame by frame trying to force the unforced state sequence to become closer to the desired state sequence. This is carried out by using the batch version of the LVQ algorithm [4]

$$\mathbf{m}_i = \frac{\sum_j s_j \mathbf{x}_j}{\sum_j s_j}, \tag{3}$$

where s_j is the indicator of the correctness of the classification, +1 in case of correct classification and -1 otherwise. If the denominator $\sum_j s_j$ is zero or negative, no updating is done.

Each monophone was modeled by a five-state left-to-right HMM. Silence was modeled using a single-state HMM. Each state of the monophone HMM was modeled by a codebook with 20 kernels. This resulted in altogether 2420 kernels. Each diphone was also modeled using a five-state left-to-right HMM. The pdf of each diphone model state was modeled by ten kernels resulting in 20300 kernels. Monophones and diphones were trained in a similar manner using the training procedure described above. After SOM initialization, five batch rounds of SKM training were performed. Monophone models were then fine tuned with five LVQ-batch rounds. In case of the diphone models, after SKM training before applying the LVQ, those kernels were removed whose mixture weight coefficients were less than 0.05. The number of the remaining kernels was then 14624. This pruning did not affect the recognition accuracy using development data. Two rounds of batch LVQ were then applied to the pruned diphone models.

The transition probabilities between diphone models were smoothened using a small constant for unseen diphone pairs. The diphone models themselves were trained only for those monophone pairs which were present in the training data. This resulted in 406 diphone models the number of monophone models being 25 including the silence model.

4 Rcognition results

The purpose of the first experiment was finding out the difference between monophone models and diphone models in terms of the recognition accuracy. The results for training, development, and test data are shown in Table 1.

The next question was how to achieve the recognition accuracy of the diphone models with less computation. In the current approach the solution was to use a two-level recognition system where the result of the monophone recognition of each input utterance is utilized for selecting a subset of the diphone models to be used in the next-level of the recognition. Those diphones were selected whose left or right phoneme contexts were present in the recognized monophone sequence. Besides allowing the correction of phoneme substitution errors, this enables also the correction of single-occurring deletion or insertion errors in the result of the monophone recognizer.

The recognition results using a two-level system are shown in Table 2. Since almost identical recognition accuracies were obtained when using all diphone models and when ignoring those diphone models whose left or right contexts were not detected by the monophone models, the results can be considered very satisfactory.

Table 1: *Speaker-independent phoneme recognition using monophone and diphone models. Errors per cent.*

data set	monophone models	diphone models
training	20.3	11.9
development	29.0	22.9
test	27.5	20.7

Table 2: *Speaker-independent phoneme recognition using the proposed two-level system, errors per cent. Full set of diphones consisted of 406 models.*

data set	two-level recognizer	average number of evaluated diphone models per word utterance
training	12.3	223
development	23.1	229
test	20.8	222

5 Model clustering

In the experiments described above, monophones were used for selecting the diphone candidates for being evaluated in the second level of the recognition. Alternatively, also clustered models can be used. In order to apply clustering methods, a distance or similarity measure must be defined for the models. Here the distance between two densities $f(x)$ and $g(x)$ was computed as

$$\int |f(x) - g(x)|^2 dx = \int f(x)^2 - 2f(x)g(x) + g(x)^2 dx. \tag{4}$$

Since $f(x)$ and $g(x)$ were in this work Gaussian mixture densities, the integrand consists of the sum whose

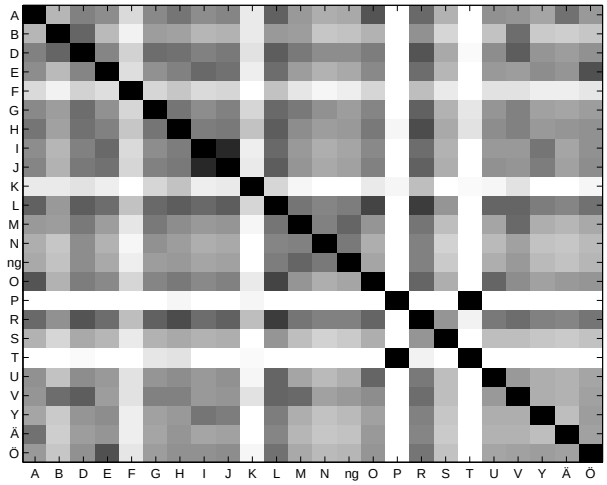

Figure 2: *Pairwise distances between monophone models. Dark shade of gray indicates small distance and light shade large distance, respectively.*

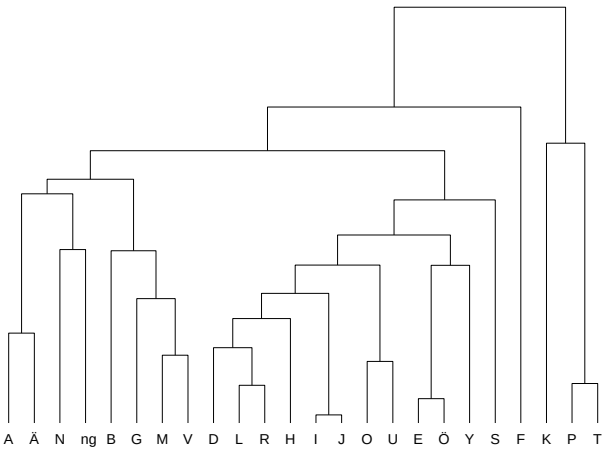

Figure 3: *Hierarchical clustering of monophone models according to the pairwise distances shown in Fig. 2.*

all elements are products of two Gaussians. The product of two Gaussians is a Gaussian itself and since its integral is easy to compute, also the distance between two Gaussian mixture densities can be easily computed analytically.

An example of the pairwise model distances between monophones is shown in Fig. 2. The pairwise distances between the pdfs of the states of the 5-state HMMs were summed in order to get the distance between HMMs.

Based on the distances between HMMs, an agglomerative clustering procedure was performed [8]. In the beginning of the clustering all models were in separate clusters. The clusters were then merged one by one, see an example in Fig. 3. The distance between two clusters was defined to be the maximum distance between the members of the clusters. In the recognition system, the first-level recognizer consisted now of clustered models. In the second level those diphone models were evaluated who were members of the recognized clusters. Both clustered monophone models and clustered diphone models were experimented. But since the recognition results using unclustered monophone and diphone models described in the previous section were so close to the accuracy using the full set of diphone models, the clustering did not give any improvement to these results.

6 Conclusions

The starting point of this work was that specific and accurate acoustic models are usually computationally more time-consuming. In order to decrease the amount of computation, the recognition system can start using coarse models and then proceed with a subset of more detailed models. In this work a two-level phoneme recognition system was constructed where the output of the monophone recognizer was used for selecting a subset of diphone models for being evaluated. Those diphone models were selected whose left or right contexts were present in the recognized monophone sequence. The results were very satisfactory, only half of the diphone models needed to be evaluated on average in the second level of the recognition for each input utterance without significantly deteriorating the recognition accuracy. The experimented approach is very straightforward and it can be easily applied to triphones or other higher order acoustic N-grams.

References

[1] Holmström, L. and Hämäläinen, A., "The self-organized reduced kernel density estimator", Proceedings of International Conference on Neural Networks (ICNN), pp. 417–421, 1993.

[2] Kohonen, T., "The Self-Organizing Map", Proceedings of the IEEE, 78(9):1464—1480, 1990.

[3] Kohonen, T., Self-Organizing Maps, Springer, 1995.

[4] Kohonen, T., "Self-Organizing Maps of Symbol Strings", Technical Report A42, Helsinki University of Technology, Laboratory of Computer and Information Science, 1996.

[5] Kurimo, M., "Training Mixture Density HMMs with SOM and LVQ", Computer Speech and Language 11(4):321–343, 1997.

[6] Ney, H. and Ortmanns, S., "Progress in Dynamic Search for LVSCR", Proceedings of the IEEE, 88(8):1224–1240, 2000.

[7] Rabiner, L.R., "A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition", Proceedings of the IEEE, 77(2):257–286, 1989.

[8] Young, S.J. and Woodland, P.C., "The Use of State Tying in Continuous Speech Recognition", Proceedings of the 3rd European Conference on Speech Communication and Technology (Eurospeech'93), pp. 2203–2206, 1993.