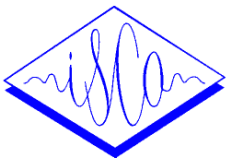

3rd European Conference on
Speech Communication and Technology
EUROSPEECH'93
Berlin, Germany, September 19-23, 1993

OPTIMIZATION OF AN HMM - BASED CONTINUOUS SPEECH RECOGNIZER

F. Class A. Kaltenmeier P. Regel-Brietzmann
Daimler-Benz AG, Research Institute, 7900 Ulm, Germany

ABSTRACT

This paper describes some optimizations to our speech recognition system, which is based on semi-continuous Hidden Markov Models (SCHMM) of subword units. The optimizations pertain to codebook generation, Linear Discriminant Analysis (LDA), initialized training, and definition of subword units. The recognition rate of the continuous version of the system increased from 79% to 95% combining all of the optimization steps.

Keywords: SCHMM, LDA, continuous speech recognition, codebook generation, subword units.

1 INTRODUCTION

The requirements of speech recognition systems have increased rapidly during the past few years, with respect to higher recognition accuracy and flexibility of the application as well. We therefore developed a modular system with a uniform architecture to handle all the different tasks such as flexible vocabulary, speaker–adaptive or speaker–independent recognition, isolated words or continuous speech recognition and word or structure spotting, respectively. The system is based on Hidden Markov Models (HMM) of subword units.
Several modules of this system have been improved and optimized, so that we have obtained a significant increase in the total performance. All improvements are based on algorithmic methods, not on heuristics. The optimizations reported on in this paper have to do with feature normalization and transformation, codebook and HMM training, and definition of subword units.
In Section 2 we give a detailed overview of our system, and in Section 3 the optimization steps are described, together with the improvements obtained by each optimization.

*This work was partly supported by the German Federal Ministry for Research and Technology (BMFT) under Grant No. 01 IV 102 E. The authors are solely responsible for the contents of this publication.

2 SYSTEM OVERVIEW

The on–line part of the system consists of feature extraction and transformation, vector quantization, and HMM verification, whereas lexicon and HMM parameters are generated off–line.
We use mel–based cepstral features with a centisecond frame rate (for details see [6]). A feature vector consists of 11 coefficients: 10 cepstral coefficients $c_1 \cdots c_{10}$ and 1 log energy component.
The cepstral features are then normalized to remove spectral influences of the speaker, microphone, room acoustics, and transmission line. It has been shown in [3] that the long–term spectrum of speech can be characterized by the mean values of cepstral features. This fact was used to develop a method for continuously adapting the system to varying spectral influences. The algorithm is quite simple: The mean values of cepstral feature vector coefficients are estimated frame by frame using an exponential window and then subtracted from an actual cepstral vector. This normalization — high-pass filtering — can be thought of as an unsupervised speaker and system adaptation. The choice of the time constants of the exponential window determines the adaptation speed.
Furthermore, in the non–optimized system the Δ and $\Delta\Delta$ features are computed from the (normalized) cepstral coefficients and the log energy contour as well, using a 5–frame window. This yields a total number of 33 parameters per frame.
Optionally we can activate a supervised speaker adaptation, which is based on feature vector transformation ([1]–[3]).

Each feature vector is then vector quantized in the vector quantization stage. Since we use semi–continuous HMMs with Gaussian distributions (full covariance matrices) in our recognition system, we create (in the non–optimized system) multiple codebooks for cepstral coefficients, log energy, Δ, and $\Delta\Delta$ coefficients assuming statistical independency. These 4 codebooks consist of 256 Gaussian distributions for cepstral coefficients, Δ, and $\Delta\Delta$ coefficients, and of 64 Gaussian distributions for the energy. The vector quantization is a classification process on all of the Gaussian distributions of all codebooks. Since we use a soft–decision vector quantization, the output of our vector quantizer is a couple of likelihoods (normalized densities) for each input feature vector.
The output sequence is then verified in the HMM verification module with the Viterbi Algorithm. For this verification process we need two sets of parameters: the coefficients of the HMMs (emission and transition probabilities) and a lexicon.
The lexicon is generated off–line and implemented in a tree structure (beam search) in order to accelerate (in conjunction with an efficient pruning technique) the recognition process. Furthermore, a bigram language model can be optionally integrated into the recognition process.
Words are implemented in the lexicon as a concatenation of subword units. These subword-unit chains are composed using the basic phonetic units out of the phonetic description of the words and then transforming the phonetic units into the subword units by a set of rules.
Definition of the subword units and structure of the HMMs is closely correlated. Since we use context–dependent diphones, we take three–state HMMs with a loop for each state and a one–state skip. The training of the HMM parameters is carried out using the foreward/backward algorithm.
The output of the HMM verification process depends on the application. For an isolated word task the output will be a list of word hypotheses, ordered according to their probabilities.
For applications with a limited vocabulary (less than 1000 words) we developed an algorithm to determine the N best sentences. Options of this algorithm are the possibility of implementing bigram language models (in conjunction with the tree lexicon) and its ability of word or structure spotting, respectively.
Continuous speech applications with large vocabularies require a powerful linguistic postprocessing. Adapted to such a postprocessing the output of our continuous speech recognizer is a word hypothesis graph, the arcs of which are word hypotheses together with acoustic scores. A special error criterion for assessment of such graphs is introduced which is accepted as standard in the German speech community. The error rate is a function of the amount of word hypotheses in the graph (for details see [6]).
The advantage of presenting the recognition result this way is that this graph implicitely contains a large number of sentences, out of which the linguistic postprocessing can find the best one with respect to syntactic, semantic, and pragmatic aspects.

3 OPTIMIZATION STEPS

The optimization steps described here are hierarchical and cannot be considered independently. They are concerned with various modules of the system, such as normalization and transformation of the feature vectors, estimation of HMM parameters (training), and definition of subword units.
The optimizations have been tested on a continuous speech recognizer, which was trained on a database of 200 phonetically balanced sentences, spoken by 20 speakers (total 4000 utterances). The test set out of a 'traintable information' application was spoken by 20 different speakers, each of whom uttered 10 different sentences (total 200 utterances). The test vocabulary consisted of 1070 words with a coverage of less than 15% with the training vocabulary. Training and test conditions are described in detail in our companion paper [6]. The error rates reported below are related to 100 word hypotheses per second.

3.1 Two–Stage Codebook Generation

A commonly used method for codebook generation is the LBG Algorithm. Normally Euclidian distances are used. Codebooks are iteratively estimated for N levels, i.e. we start at level 0 and obtain an N–bit codebook (codebook size 2^N) running 10 iterations at each level using Euclidian distances. During the last iteration at the N–th level the covariances of each codebook cell are estimated.
In a second step a fine tuning of the N–bit codebook is carried out by 4 iterations using Gaussian densities with full covariances as a distance measure. By this Gaussian modelling a better group-

ing of the feature space will be obtained. The error rate was reduced by about 2% compared to the simple Euclidian distance codebook generation.

3.2 Linear Discriminant Analysis

The *Linear Discriminant Analysis* (LDA) is an efficient method for separating different classes in the feature space [5]. Whereas LDA is commonly used separately for each feature set, we tried to apply just one LDA to all feature sets. The basic (and new) idea is to combine several feature vectors (cepstral coefficients + energy) into one feature vector as input to the LDA. This type of time window implicitely includes all of the temporal dynamic effects (Δ-, $\Delta\Delta$- $\cdots$ features) of the speech signal.
For LDA, 9 frame vectors are combined to one 9·11=99-dimensional feature vector which is transformed into 32 coefficients by the LDA transform matrix. In order to compute the LDA transform matrix, the class–specific covariance matrizes are needed. The problem of implementing LDA in an HMM recognizer based on subword units is defining the classes and computing the corresponding covariance matrices. In our case, classes correspond directly to SCHMM states or state clusters of subword units. State–dependent covariance matrices are computed during the last iteration of a first forward/backward training. This first training pass is based on multiple codebooks for cepstral, Δ, and $\Delta\Delta$ coefficients whereas the second pass is solely based on *one* codebook with 32–dimensional gausssian distributions of LDA–transformed features. The complete training of the system is described in the next section. We obtained a remarkable decrease of the error rate by LDA of about 5%, compared to the 4 codebook version.

3.3 Multi–Stage Training

Figure 1 gives an overview of the complete system training. The inputs to the training are a lexicon with tree structure and mean–value normalized training samples. The first training stage is the two–step codebook generation (see Section 3.1). All of the training samples are then vector quantized with the four codebooks. In a first training step 37 basic models (phone models) are trained using the Foreward/Backward (FB) Algorithm (as in all of the following training steps). In the next step 128 diphone models are initialized with the parameters of the 37 models and then trained with 10 iterations of the FB algorithm. During the last iteration we must compute the state-dependent means and covariances, as mentioned above. This is achieved as follows: For the actual frame t we obtain the probability of being in state S_i, given the observation sequence **O** (sequence of codebook symbols) and the model λ. The feature vector corresponding to the frame is weighted by this probability, and the covariance of state S_i is updated by the weighted feature vector. This weighting and updating is performed for all states. As mentioned above, we use a time window of 9 frames (concatenation of 9 frames) as input to the LDA. To accomplish this we combine the feature vectors of frames $i-4$ to $i+4$ into one vector, weight its components by the probabilities, and update means and covariances of the corresponding states. For 11–dimensional feature vectors we obtain 99–dimensional means and covariances. In this step we need not only the quantized vector, but also the cepstral data. Using these 'class–specific' means and covariances we can compute the LDA transformation matrix. The classic way for applying LDA would be to compute a new codebook using LDA–transformed training samples. We simplify this by directly transforming the 99–dimensional means and covariances by the LDA transformation matrix. We then obtain 32–dimensional means and covariances, which will be further designated as *LDA codebook*.
Now all of the training samples are LDA–transformed and vector quantized with the LDA codebook. In the third training stage the 37 phone models are trained again using LDA–transformed input data, and last in a fourth training stage the final 128 diphone models are trained, initialized by the parameters of the 37 models.
The initialization of context–dependent HMMs with pretrained context–independent phone models lowers the error rate by about 2% compared with the direct training of 128 models.
It is worth noting that the first two training steps are just to determine the state–dependent (class–specific) means and covariances.

3.4 Optimization of Subword Units

In German there are 37 phones commonly used as basic phonetic units. It is well known that

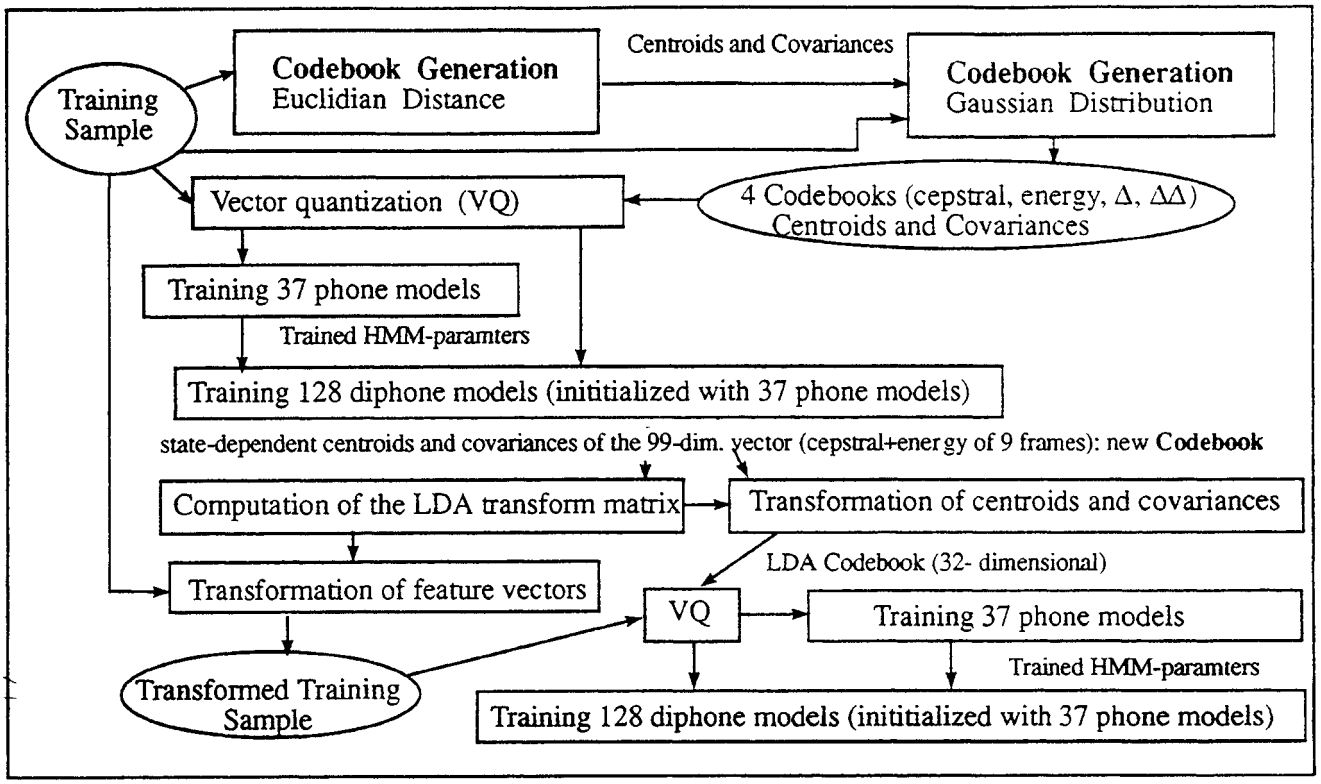

Figure 1: Overview of the complete system training.

phone transitions contain a lot of important information, so we carried out some experiments to model context–dependent diphones with HMMs. The context dependency is due solely to the training set and is independent of the test set.
Increasing the number of subword units makes sense only if there is enough training material to train all of the units in many different contexts. Taking this into account we obtained best results with 128 subword units (HMM models). The error rate decreased by 7% compared to the 37 context–independent phone models.

4 CONCLUSIONS

In this paper we have reported about several optimization steps, which have been implemented in our modular speech recognition system. The system is based on semi–continuous HMMs of subword units. The optimizations involve an improved codebook generation procedure, the introduction of a modified version of Linear Discriminant Analysis, a multi–stage training, and definition of subword units. Experiments were carried out in the continuous mode using a word hypothesis graph as output. Each of the optimization steps improved the performance of the system, but by combining all of the optimizations we increased the recognition rate from 79% to 95%.

References

[1] F. Class, A. Kaltenmeier, P. Regel, K. Trottler: *Fast Speaker Adaptation for Speech Recognition Systems.* ICASSP '90, Albuquerque, April 90, pp. 133–136.

[2] F. Class, A. Kaltenmeier, P. Regel, K. Trottler: *Fast Speaker Adaptation Combined With Soft Vector Quantization In An HMM Speech Recognition System* ICASSP '92, San Francisco, April 92, pp. 1-161 ... 1-164.

[3] F. Class: *Standardisierung von Sprachmustern durch vokabular-invariante Abbildungen zur Anpassung an Spracherkennungssysteme.* Fortschrittberichte VDI, Reihe 10, Nr.131; VDI-Verlag Düsseldorf.

[4] X. D. Huang, M. A. Jack: *Semi-continuous hidden Markov models for speech signals.* Computer Speech and Language, Vol. 3, 1989, pp. 239–251.

[5] K. Fukunaga: *Introduction to Statistical Pattern Recognition.* Academic Press, New York, Second Edition, 1990.

[6] F. Class, A. Kaltenmeier, P. Regel: *Evaluation of an HMM Speech Recognizer with Various Continuous Speech Databases,* these proceedings.